Universitat de Lleida

TREBALL FINAL DE GRAU

Estudiant: Nadal Rivero, Arnau
Oliveros Muelas, Miquel

Titulació: Grau en Enginyeria Informàtica

Títol de Treball Final de Grau: **Preparació de l'espai de treball en un dispositiu ocular emprat per la detecció d'objectes per analitzar l'Awareness.**

Director/a: Garrido Navarro, Juan Enrique
Granollers Saltiveri, Antoni

Presentació

Mes: Octubre

Any: 2020

Resum

Els avenços tecnològics de la història sempre han estat causats per la idea de l'ésser humà de realitzar coses fora de l'abast de la seva mà com per exemple: comunicar diverses persones de punta a punta del planeta o portar un cotxe sense haver de conduir-lo.

Una d'aquestes idees és poder controlar completament el nostre entorn, sense perdre el més mínim detall d'aquest. Això ha dut a l'aparició del terme *Awareness*, que es refereix al fet que un usuari sigui conscient del que està passant al seu entorn, l'involucri directament o no.

En aquest treball, es pretén implementar una aplicació de detecció d'objectes que ens permeti analitzar i estudiar com l'awareness pot ajudar l'usuari en aquest tipus d'aplicació. Segons els resultats obtinguts es determinarà si suposa un avenç i una millora en el camp de l'awareness de les persones, aportant informació del seu entorn que sigui realment d'ajuda per al desenvolupament de certes activitats en el context adient.

PARAULES CLAU: Awareness, Detecció d'Objectes, Android, smartglasses, NUIs, Realitat Mixta

Abstract

The technological advances of history have always been given by the idea of the human being to make things out of their reach, for example: to communicate several people from one end of the planet to the other or to take a car without having to drive it.

One of these ideas is to be able to completely control our environment, without losing the slightest detail of it. This has led to the emergence of the term *Awareness*, which refers to a user being aware of what is happening to their environment, whether it involves them directly or not.

In this work, we intend to implement an object recognition application that allows us to analyze and study how awareness can help the user in this type of application. According to the results obtained, it will be determined if it is an advance and an improvement in the field of people's awareness, providing information from their environment that is really helpful for the development of certain activities in the right context.

KEYWORDS: Awareness, Object Detection, Android, smartglasses, NUIs, Mixed Reality

Índex

Índex de Figures

Agraïments

Per la realització d'aquest treball volem donar les gràcies pel suport rebut a professors, amics i familiars.

Volem agrair de forma especial a l'empresa Invelon i el seu equip tot el suport que ens ha ofert, tant per proporcionar-nos els dispositius oculars com per guiar-nos durant el projecte, sense ells hagués estat impossible realitzar el treball.

Als nostres tutors de treball de final de grau, Toni Granollers i Juan Enrique Garrido, per inspirar-nos, donar-nos tot el suport necessari i estar allà sempre que els hem necessitat. També per ser uns grans tutors, implicats al cent per cent amb nosaltres i donar-nos tots els recursos necessaris sempre que han sigut necessaris.

Sumar l'espai ofert pel Grup de Recerca en Interacció Persona Ordinador i Integració de Dades (GRIHO) per poder realitzar el treball a la seva sala de treball.

Aquest no podia faltar, als familiars i amics per donar-nos una empenta quan ho hem necessitat i donar-nos ànims quan no aconseguíem fer possibles els nostres objectius.

1. Motivació

La motivació per dur a terme aquest treball és deguda a diferents factors. El primer d'ells i més important: la proposta per part dels actuals tutors de TFG per realitzar un treball relacionat en l'àmbit de l'awareness i de la IPO (Interacció persona-ordinador). El concepte de treball ens va agradar i vam decidir realitzar-lo.

En segon lloc, una altra motivació que ens va fer decidir va ser l'oportunitat de poder treballar i experimentar amb dispositius oculars. El fet de ser una tecnologia poc visible, o millor dit, poc accessible a causa de la inexistència d'unitats per adquirir-les, va fer que tinguéssim més ganes de realitzar aquest treball.

A més, l'oportunitat de poder explorar nova tecnologia ens aporta un al · licient extra per la realització del treball.

Amb aquests dos factors combinats, obtenim un resultat on, per una part, creiem que el tema és poc explorat als treballs actuals de recerca i podem aplicar aquest concepte a una tecnologia amb un sentit concret.

2. Introducció

L'awareness el podríem definir com la qualitat o estat de ser conscient: coneixement i comprensió que alguna cosa està succeint o existeix[1]. És un concepte molt ampli que suposa tot un repte si es vol controlar. Mitjançant la informàtica es poden realitzar diverses solucions que permetin ajudar en l'awareness de les persones. En aquest treball proposem un sistema de detecció d'objectes com a solució aplicable per estudiar com l'awareness pot ajudar en aquest tipus d'entorns als usuaris. Però, pot realment suposar una millora per a l'awareness de les persones?

La línia de treball definida es basa a oferir un dispositiu amb una aplicació de detecció d'objectes emmarcada en un context determinat per primerament, analitzar com influeix l'awareness d'una persona en un entorn determinat i finalment millorar aquesta solució en base als resultats extrets. Arran d'aquest treball es pretén fer l'estudi pertinent centrat en la real utilitat i benefici que suposi aquest sistema en el context escollit.

La línia de treball seguida ha derivat en un treball més concret que podríem considerar com la primera fase del treball general proposat en un primer moment. Aquesta primera fase tracta de crear un entorn aplicable per a l'estudi objectiu: una aplicació executable en un dispositiu Android, concretament les smartglasses de Vuzix, capaç d'integrar OpenCV per aconseguir l'objectiu de detectar objectes. En aquest treball s'aborda aquesta primera fase, així doncs, **l'objectiu directe del treball és realitzar una aplicació de detecció d'objectes**.

La primera fase es va iniciar amb la cerca de smartglasses (de tota mena de realitats) que ens permetessin disposar de l'entorn requerit per la realització del nostre estudi. La cerca de dispositius va ocupar una gran part del treball a causa de limitacions de pressupost, falta d'eines per desenvolupar aplicacions i disponibilitat en el mercat actual, que presentaven algunes opcions; això ens van obligar a desestimar opcions molt interessants i a dedicar més temps del desitjat en trobar el dispositiu adient. Tot plegat, deriva en una col · laboració amb l'empresa Invelon[1] la qual ens proporciona les ulleres Vuzix M300XL i posteriorment les Vuzix M400.

Arran d'aquesta col · laboració, s'adapten els objectius principals del treball dins de la línia de treball prèviament planificada segons els interessos de l'empresa col · laboradora, on el primer objectiu directe és crear una aplicació per a Android de detecció d'objectes funcional que utilitzi OpenCV per a la vessant de visió per a computadors. Aquesta col · laboració permet seguir amb la planificació inicial i aporta coherència per establir unes bases solides d'implementació.

Arribats en aquest punt de la fase, es van iniciar les tasques de desenvolupament de l'aplicació, les quals van suposar fer un estudi previ exhaustiu de la llibreria OpenCV i de quines solucions es podien desenvolupar amb la seva utilització en relació amb la

[1]Invelon Technologies és una empresa tecnològica situada a Lleida que desenvolupa solucions de Realitat Virtual i Realitat Augmentada. Pàgina web: `https://www.invelon.com/`

detecció d'objectes. Això ens va portar al descobriment i posterior estudi de la xarxa neuronal Darknet i de l'algorisme YOLO.

Durant el desenvolupament de l'aplicació es varen trobar diferents complicacions a causa de l'aparició del virus COVID-19. Aquesta situació frena per complet el desenvolupament i, per tant, agreuja els problemes principals que s'havien presentat, afegir la llibreria en un projecte Android així com no poder fer proves amb el dispositiu, ja que, aquest es va quedar a la sala GRIHO[2]. Un cop comença la desescalada del confinament, es recupera el dispositiu i ja s'inicia la prova d'integració amb èxit: execució d'una aplicació de reconeixement d'objectes amb OpenCV a les Vuzix per al control de la càmera del dispositiu i tractament dels frames obtinguts.

Un cop aconseguida la integració i obtenció correcta dels frames de la imatge en viu de la càmera, es procedeix a integrar la xarxa neuronal Darknet per tal de començar a reconèixer objectes generals (com pot ser una cadira, una persona o una pantalla, entre altres) mitjançant la utilització de models prèviament entrenats de l'algorisme YOLO. El model aquí utilitzat s'obté de la pròpia pàgina web de Darknet[3] i estan entrenats amb la base de dades de COCO, de la qual n'obtenim una llista de les diferents classes d'objectes que utilitza el model.

Seguidament, es procedeix a la millora del rendiment de l'aplicació, ja que el càlcul seqüencial de la detecció d'objectes redueix en gran manera els frames processats per segon. La solució trobada implica realitzar la detecció d'objectes en els frames de forma paral · lela a l'obtenció d'aquests i, mitjançant el control dels sensors ambientals, millorar i controlar els càlculs de frames no necessaris.

Tot i millorar el rendiment, la capacitat de còmput de les ulleres era reduït. Invelon, com a solució a aquesta situació, ens ofereix un model de les Vuzix més actual i, per tant, amb més capacitat computacional. Gràcies al nou dispositiu, les Vuzix M400, s'executa l'aplicació de forma correcta i sense gairebé pèrdues de frames. Tot i això, la part final de la fase es dedica a buscar més solucions, encara que no han estat aplicades, per a la millora del rendiment de l'aplicació en termes del càlcul innecessari de frames.

Tal com es comenta a l'inici, la línia de treball es divideix en tres fases. Aquestes dues restants no s'han pogut dur a terme per falta de temps i previsió de la complexitat del treball. El treball es va plantejar amb un objectiu complex i això explica el perquè s'ha arribat a fer la implementació i no s'ha pogut estudiar l'awareness, però, s'ha deixat una base sòlida i pautada per seguir amb el treball. A continuació, es deixarà plasmada la idea del treball inicial per entendre on hauria de derivar aquest procés general.

A la segona fase del projecte, es preveu fer un estudi sobre els beneficis que pot aportar

[2]Grup de Recerca en Interacció Persona Ordinador i Integració de Dades de la Universitat de Lleida.

[3]Es pot obtenir mitjançant els enllaços que es troben a la pàgina web:`https://pjreddie.com/darknet/yolo/`

amb el seu awareness l'ús de l'aplicació creada en un context determinat. Per dur a terme això, en primer lloc, s'hauria d'escollir un context on aplicar la tecnologia de detecció d'objectes com a eina per millorar l'awareness del usuari. Amb la integració d'awareness estem proposant fer un ús d'aquesta tecnologia i les smartglasses per oferir informació d'allò que envolta a l'usuari d'una forma determinada. La informació oferta a l'usuari dependrà del context on s'apliqui, però aquesta informació tractarà d'oferir a l'usuari: canvis en el seu entorn (objectes, persones, espais, etc.), localitzacions, situació de l'usuari, etc.; en qualsevol cas, quelcom que es pugui oferir amb l'eina de detecció d'objectes.

Un cop s'ha determinat el context que se li vol donar a l'aplicació i com aplicar-ho, s'haurà d'estudiar com l'usuari interactuarà (si és necessari) amb l'aplicació, ja sigui de forma directa, per veu o amb altres interaccions segons les necessitats.

Definit el comportament de l'aplicació i la solució que ha d'aportar en el context escollit, aquesta ha de ser ficada a prova en un treball de camp en el qual s'avaluarà la real utilitat/avantatge que suposa per a l'usuari portador que es trobi en el context escollit.

Finalment, com a última, però no menys important, tenim la tercera fase, on s'haurà de desenvolupar una solució per resoldre la portabilitat d'aquesta aplicació a altres plataformes o altres models de smartglasses. Aquesta fase es podria dur a terme paral · lament amb la segona, ja que no depenen una de l'altra. El problema principal que es trobarà és que tots els sensors utilitzats per millorar el rendiment i el tractament de frames emprat està fet a mida per les Vuzix M300XL i M400. La solució plantejada i més viable és que a l'inici de l'execució de l'aplicació Android existeixi una eina que permeti de forma automàtica reconèixer el dispositiu i adequar els valors necessaris, com per exemple la resolució de la càmera. En cas de voler traslladar a una altra plataforma que no sigui Android, s'hauria de tornar a la primera fase del treball i investigar com aplicar OpenCV a altres plataformes, ja que, en aquest treball, se centra tot en Android.

3. Objectius del Treball

1. Estudiar la tecnologia de Detecció d'Objectes.
    a) Entrar en contacte amb eines de desenvolupament i utilització d'aquest tipus de tecnologies (p.e llenguatges de programació, llibreries o plataformes).
    b) Adquirir coneixement del mercat amb relació a les diferents solucions per a la Detecció d'Objectes.
2. Preparar el dispositiu.
    a) Integrar en un dispositiu de ulleres de realitat mixta una aplicació per sistemes operatius Android.
3. Desenvolupar una aplicació funcional.
    a) Integrar OpenCV com llibreria de software de visió per computador.
    b) Detecció d'objectes en el conjunt de l'aplicació amb OpenCV.
    c) Reduir la càrrega computacional de l'aplicació.
4. Establir planificació per al posterior estudi del Awareness.
    a) Consolidar l'aplicació creada com a base per a aquest estudi.

4. Estat de l'Art

En aquest apartat es posarà en context i s'explicaran els conceptes necessaris i indispensables per entendre als apartats següents, sense els quals, no s'entendran molts temes que es tractaran.

4.1. Awareness

L'Awareness és l'objectiu principal de la línia general del projecte i, tot i que no es realitzi l'estudi en aquesta fase, és necessari explicar-lo i definir-lo.

El concepte awareness des del punt de vista del camp de la computació i de la recerca podria definir-se de la següent manera:

"*Percepció que l'usuari té del que la resta d'usuaris fan en el sistema*"

Una definició molt pròxima és la que va oferir Endsley[2], indicant que awareness és

"*knowing what is going on*",

és a dir, saber què està passant. D'una manera més completa s'amplia la visió del concepte en:

"*Awareness involucra conèixer qui està al voltant, quines activitats estan ocorrent, qui està parlant amb qui; això proporciona un punt de vista d'un altre usuari en l'entorn diari de treball. Awareness ha de conduir cap a interaccions informals, connexions espontànies i el desenvolupament de cultures compartides.*"

Carl Gutwin i Saul Greenberg resumeixen les quatre característiques bàsiques del awareness recollides d'altres treballs anteriors, permetent completar la visió del concepte:

- Awareness és el coneixement sobre l'estat d'un entorn particular.
- Els entorns canvien amb el temps, amb el que el awareness ha d'actualitzar-se constantment.
- La gent manté la seva awareness interactuant amb el sistema.
- El awareness és un objectiu secundari, la meta és dur a terme unes certes tasques en el sistema, no que el sistema mantingui el awareness simplement. D'aquesta manera, podem veure el awareness com un requisit no funcional del sistema.

4.2. Estils i paradigmes d'Interacció

En aquest projecte, s'incorpora un tipus de tecnologia, la detecció d'objectes amb smartglasses, que suposa un tipus d'interacció entre el dispositiu seleccionat, l'entorn i l'usuari, i representa uns estils i paradigmes que són necessaris d'entendre.

Definim com interacció tots els canvis que succeeixen entre una persona i un ordinador (Beacher Buxton, 87).

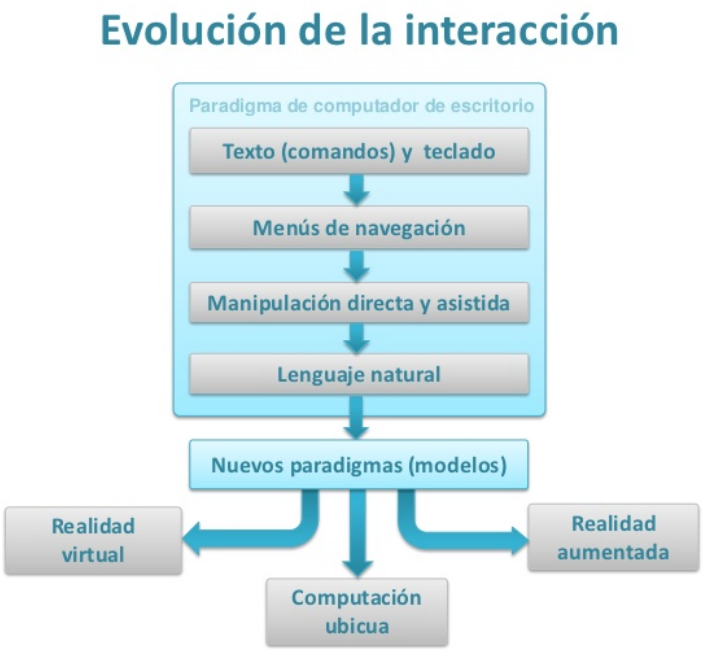

Figura 1. Evolució de la interacció entre usuari i computador.

Per una banda l'estil de comunicació és la forma en la qual els usuaris es comuniquen o interaccionen amb un ordinador. Les interaccions principals són:

- **Interfície per línia d'ordres**: donar ordes directament a un ordinador.
- **Menús de navegació**: conjunt d'opcions visualitzades a la pantalla on la selecció d'una o varies d'elles suposa l'execució d'una ordre subjacent (normalment suposa un canvi a la interfície).
- **Manipulació directa i Interacció assistida**
  - **Manipulació directa**: sintaxi d'ordres mitjançant objectes i accions.
  - **Interacció assistida**: metàfora d'un assistent/agent personal que col · labora amb l'usuari actuant en resposta a les accions de l'usuari.
- **Llenguatge natural**: interacció mitjançant la veu i el moviment.

D'altra banda, els paradigmes són els models derivats o abstraccions dels estils d'interacció agrupats per característiques similars (exemple il · lustrat a la Figura 2). Existeixen els següents paradigmes:

(a) Ordinador de sobretaula: interacció que té l'usuari amb un ordinador aïllat de l'entorn amb interfícies de manipulació directa.

(b) **Realitat Virtual**: simulació feta per ordinador que ens abstreu del món real amb una sensació que és real. La interacció és en temps real per donar efecte d'immersió.

(c) **Computació Ubiqua**: estén la capacitat computacional a l'entorn de l'usuari. No té una única localitat, distribució de molts dispositius[4].

(d) **Realitat Augmentada**: reducció de les interaccions amb l'ordinador utilitzant la informació de l'entorn com entrada implícita.

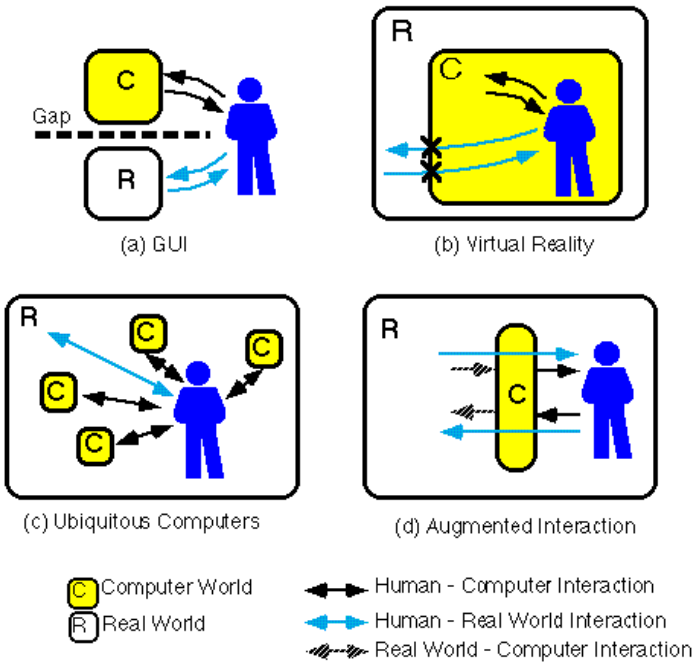

Figura 2. Paradigmes d'interacció.

4.3. Realitats aplicades a tecnologia

Amb l'evolució de la tecnologia durant el principi del segle XXI, s'ha donat un salt qualitatiu en quant a hardware i software. Lligada a aquesta evolució trobem els tipus de realitats ja que, aquestes han evolucionat i s'han adaptat a les noves tecnologies presents, com es el cas de les smartglasses. Per entendre el panorama actual, s'explicarà quin es l'estat actual d'aquest àmbit i quines realitats podem distingir.

[4]Actualment es coneix per IOT (Internet of Things).

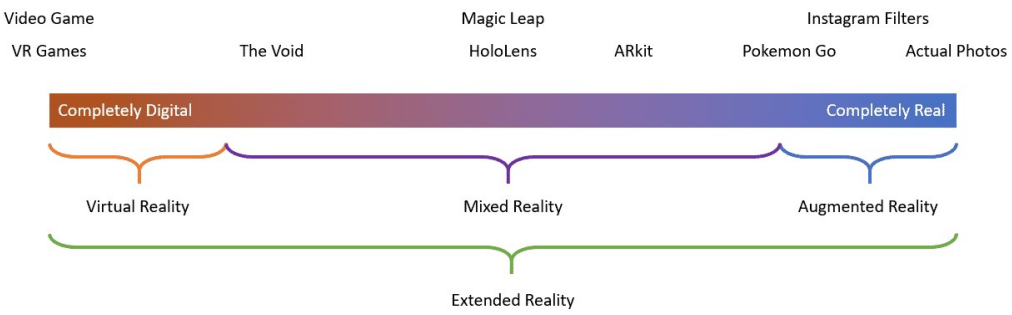

Figura 3. Espectre de realitat aplicat a tecnologia amb exemples de cada tipus.

La XR (Extended Reality) és el terme emprat per parlar de l'agregació de realitat augmentada (AR), realitat virtual (VR), realitat mixta (MR) i altres realitats "millorades" sota una definició [3]. En essència, XR cobreix tot l'espectre (Figura 3), des d'entorns "completament reals" a "completament virtuals" i també la tecnologia i els dispositius portàtils associats amb ells.

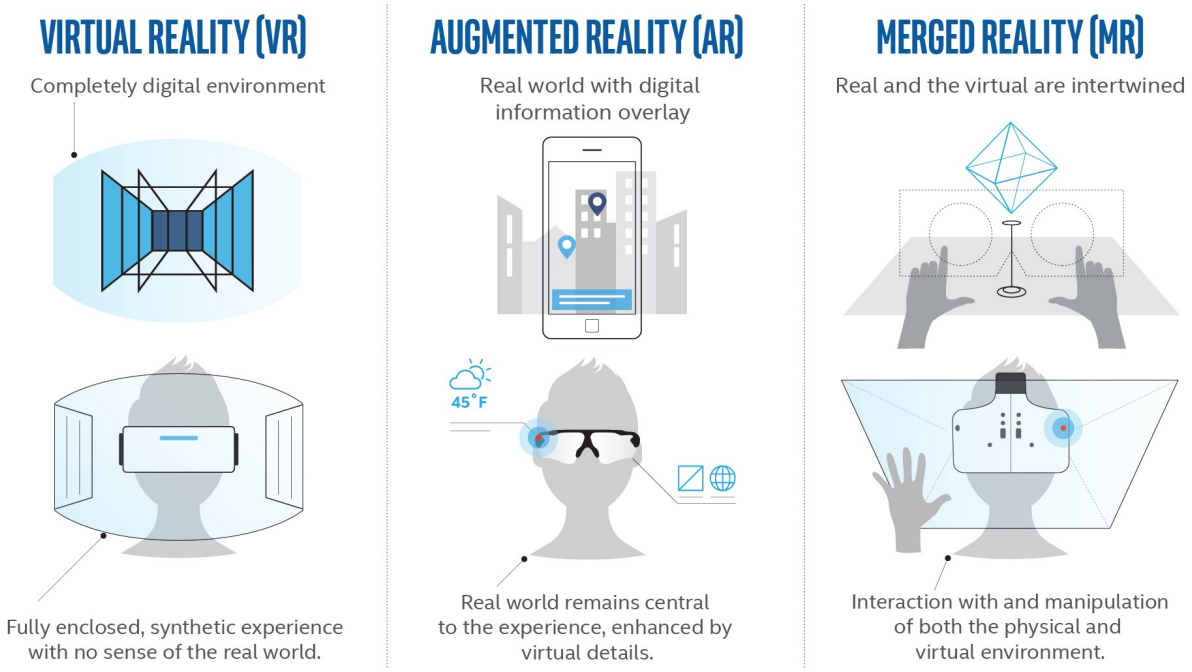

Figura 4. Esquema que resumeix les tres principals variants de realitats.

La AR (Augmented Reality) ens permet combinar elements del món virtual i el món físic. Això crea noves possibilitats per al que veiem, sentim i sentim. Aquesta tecnologia presenta el terme mitjà entre la realitat virtual i el món real, la qual cosa ens permet combinar elements de tots dos en un. La part d'augment ens permet superposar informació visual digital sobre objectes del món real.

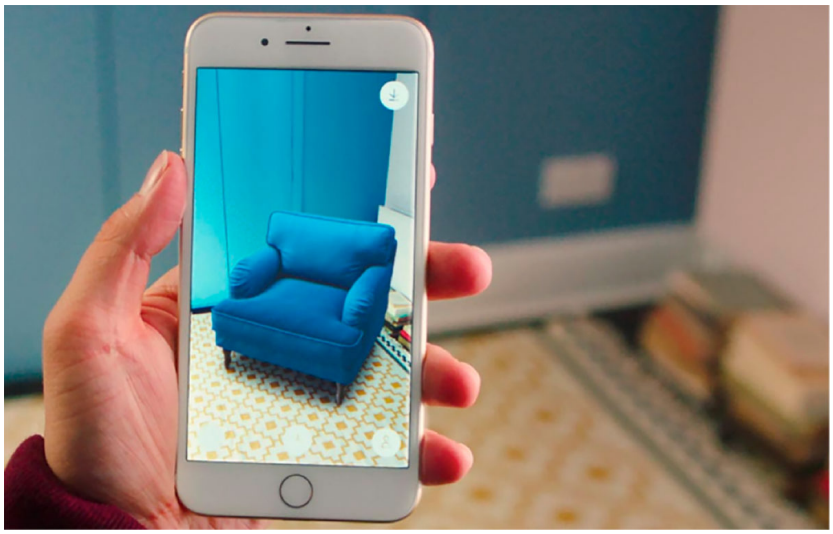

Figura 5. Exemple de AR amb una aplicació que simula mobiliari a escala real a la realitat.

La MR (Mixed Reality) és similar a la AR en la forma en què es col · loca entre la realitat virtual i la realitat física. Se la coneix comunament com una realitat híbrida, on l'objectiu és fusionar entorns físics i virtuals. És probable que en el futur s'utilitzi un terme singular per a evitar més confusió.

Figura 6. Exemple de MR on la persona rep informació de la realitat que l'envolta.

La VR (Virtual Reality) és un entorn simulat en el qual un usuari se submergeix en un entorn 3D en lloc d'utilitzar una pantalla tradicional. Tot dins d'aquest entorn és presentat per computadora a l'observador a través d'un casc de realitat virtual.

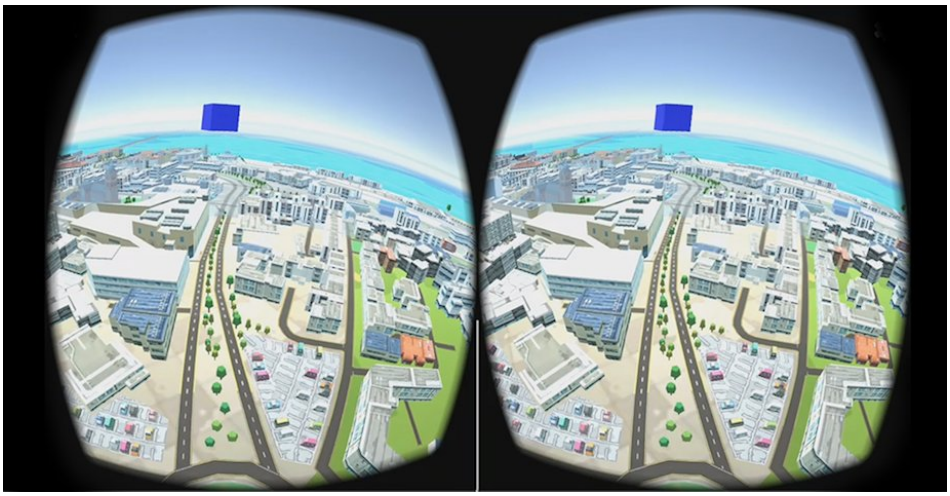

Figura 7. Exemple de VR on l'usuari esta submergit en una ciutat creada per ordinador.

4.4. *Wearables*

La tecnologia *wearable* és aquella que està dissenyada per ser vestida en el dia a dia. Qualsevol classe d'element que tingui qualsevol mena de tecnologia en ell, per més simple que sigui, s'hi pot considerar, per exemple, un rellotge digital. Tot i això, actualment es fa més referència a *wearables* a aquells dispositius electrònics que tenen microcontroladors o, dit d'una altra forma, ordinadors microscòpics dintre seu, i que, gràcies a un conjunt d'elements electrònics, sensors i software, proporcionen al portador informació addicional del seu propi cos o del seu entorn de forma immediata.

Els *wearables* tenen diverses aplicacions en múltiples àmbits: existeixen diverses solucions al mercat que permeten tenir un seguiment de la salut del portador en temps real, mostrant paràmetres tals com les calories cremades, les pulsacions del cor, el temps fent esport o vida sedentària, passes realitzades, d'entre d'altres; també hi ha les solucions d'entreteniment, aquelles que ens permeten gaudir d'una nova forma de joc, per exemple els dispositius que permeten interactuar amb un món virtual o semivirtual; després, també hi ha les solucions de productivitat, que fa referència a aquells wearables que se centren a fer més accessible certa informació a l'usuari, ja sigui com a suplement del dia a dia per a altres dispositius, o com una eina suplementaria per al treball. Com es pot veure, les solucions que aporten són molt àmplies. Tot i aquesta gran diversitat d'aplicacions, els wearables es poden classificar simplement en 6 grups [4]:

1. ***Smartwhatches*:** es tracta de rellotges intel · ligents. Permeten connectar el teu dispositiu mòbil per tal de tenir certes funcionalitats d'aquest a l'abast de la mà. Incorporen sensors que permeten, en certa manera, realitzar la mateixa funció que dispositius centrats específicament en el control de la salut.

Figura 8. Exemple d'*smartwatch*

2. ***Smart Jewellery*:** es tracta de joieria intel · ligent que, en un format tan reduït com pot ser un anell, proporciona a l'usuari informació sobre la seva salut.

Figura 9. Exemple d'*smart jewellery* - Oura smart ring

3. ***Fitness Trackers*:** monitors d'activitat que normalment prenen forma de braçalets intel · ligents. Són els dispositius que més informació proporcionen amb relació a la teva salut en l'àmbit de l'esport.

Figura 10. Exemple de *fitness tracker*

4. ***Smart Clothing*:** són peces de roba intel · ligent que, al cobrir gran part del cos, proporcionen informació de salut detallada. Actualment, és un camp que no s'ha desenvolupat gaire.

Figura 11. Exemple d'*smart clothing* - Sensoria fitness socks

5. ***Implantables*:** dispositius que prenen contacte amb l'usuari des de dintre, per exemple, amb pastilles que controlin la sang directament. És un sector encara en desenvolupament.

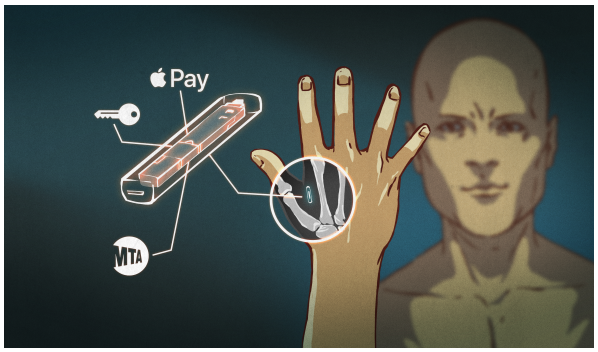

Figura 12. Exemple d'*implantables*

6. ***Head-Mounted Displays*:** dispositius que es col · loquen al cap. Aquests dispositius proporcionen a l'usuari diferents experiències de realitats alternatives (vegeu apartat 4.3) o li apropen informació d'interès a la vista, entre altres utilitats. Un exemple d'aquests dispositius en serien les *smart glasses*, que explicarem al llarg del treball.

Figura 13. Exemple d'*smart glasses* - Oculus rift (VR)

4.4.1. *Smart Glasses*

Durant generacions, el propòsit principal de les ulleres era millorar la vista. Durant les dues ultimes dècades, s'ha pogut observar un canvi accelerat en aquesta visió i s'ha afegit la intel · ligència. Aquesta última revolució ha donat un nou tipus de dispositiu anomenat "Smartglasses" o ulleres intel · ligents que es poden definir com un wearable ("ordinador corporal") ocular que ens proporciona informació juntament amb el que el usuari observa.

Actualment existeixen dos tipus principals de smartglasses: de realitat virtual i de realitat mixta. Per una banda, les ulleres de realitat virtual tenen per objectiu crear un entorn amb objectes i escenes de forma que sembli real per l'usuari. D'altra banda, les ulleres de realitat mixta tenen per objectiu superposar la realitat que veu l'usuari amb informació generada de forma multimèdia.

Les aplicacions actuals d'aquest dispositiu són molt diverses, però les podem classificar en tres sectors. En primer lloc trobem la investigació, ja que la gran majoria de aquests dispositius s'empren per desenvolupar moltes solucions de software que encara no s'han creat. En segon lloc, tenim les empreses, les quals utilitzen un hardware i software específics per problemes de qualsevol tipus, on aquest tipus de dispositiu sigui una solució òptima. Per últim, trobem el sector de videojocs on el que busquen es crear aplicacions d'entreteniment (encarat principalment a ulleres de realitat virtual).

4.5. Altres tecnologies

L'aplicació que es vol crear esta pensada per el sistema operatiu mòbil Android, desenvolupat per Google i basat principalment en un nucli Linux. Actualment es troba al mercat android 11, última versió creada i llençada el 8 de setembre de 2020.

Per al desenvolupament s'ha utilitzat l'IDE proporcionat per JetBrains i desenvolupat per Google, Android Studio, com a principal eina de desenvolupament d'aplicacions per qualsevol tipus de dispositiu mòbil que disposi d'Android com a sistema operatiu.

4.6. Machine Learning

La detecció d'objectes és una de les solucions que es poden proporcionar mitjançant la informàtica arran de l'aparició de l'àmbit del *Machine Learning*, ja que gràcies a això, un sistema és capaç de fer les deteccions i créixer en el coneixement d'aquestes sense que un humà hi prengui gaire part.

Machine Learning és una aplicació de la Intel · ligènica Artificial (IA) que s'encarrega de proporcionar a un sistema aprenentatge automàtic i la capacitat de créixer sense la necessitat de ser programat [5]. Se centra en el desenvolupament de programes que poden aprendre per si sols a partir de dades.

En conclusió, l'objectiu principal és permetre als computadors aprendre automàticament, i per fer això existeixen 4 categories o mètodes diferents de *Machine Learning*:

1. **Aprenentatge supervisat:** a partir de dades etiquetades introduïdes amb anterioritat els algorismes generen una funció per poder fer prediccions sobre la sortida de dades noves. L'entrenament d'aquest model depèn directament de les dades prèvies i la sortida que es desitja d'aquestes.

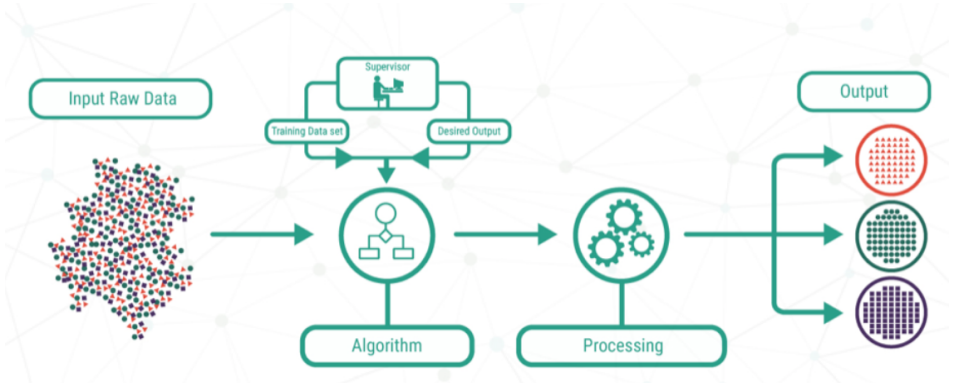

Figura 14. Aprenentatge supervisat

2. **Aprenentatge no supervisat:** La informació utilitzada per a entrenar el sistema no esta etiquetada i, per tant, aquest ha d'extreure conclusions a partir de l'anàlisi de totes les dades de les que disposa amb l'objectiu de generar estructures que li permetin, a la llarga, generar sortides correctes per les noves dades.

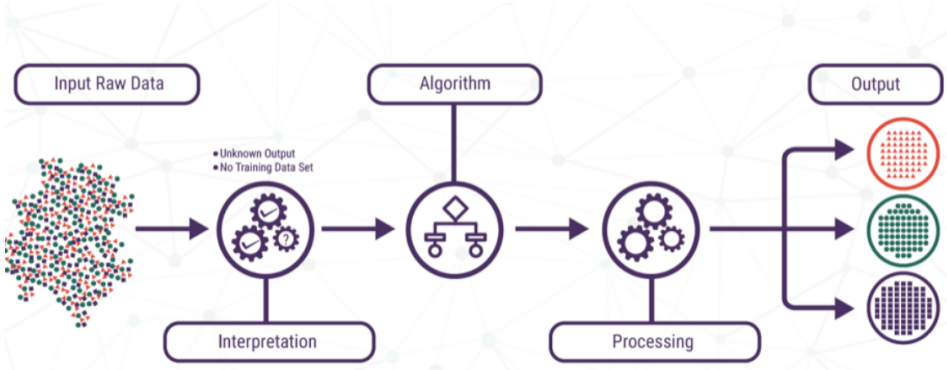

Figura 15. Aprenentatge no supervisat

3. **Aprenentatge semi-supervisat:** s'utilitza una gran quantitat de dades no etiquetades a la vegada que s'utilitza una petita quantitat d'etiquetades. L'objectiu d'aquest mètode es millorar la precisió quan les dades etiquetades requereixen de recursos qualificats i rellevants per poder aprendre d'elles. La resta de dades no etiquetades, solen no necessitar de recursos addicionals.

4. **Aprenentatge per reforç:** els algorismes estudien el seu entorn produint accions que generen errors o premis. És a dir, a partir de les dades de sortida que s'han introduït i que el sistema coneix, realitza una cerca per prova i error amb l'objectiu d'obtenir la sortida desitjada, les interaccions fetes i els resultats obtinguts per obtenir aquesta sortida s'utilitzen per a l'aprenentatge del sistema.

Figura 16. Aprenentatge per reforç

4.7. Xarxes Neuronals

Detectar objectes es realitza en una xarxa neuronal, ja que és aquest element el que conté la informació relacional necessària per a poder realitzar aquest tipus d'operacions.

Una xarxa neuronal és un conjunt d'algorismes que imiten la manera com funciona el cervell humà per a reconèixer les relacions principals entre grans quantitats de dades [6]. Així doncs, les xarxes neuronals són estructures formades per un conjunt de nodes interconnectats que es comuniquen entre si, simulant la estructura i comportament de les neurones del cervell humà. Aquests nodes s'anomenen ***perceptrons***, els quals, donades unes entrades procedents d'altres nodes amb un pes determinat, els aplica una funció d'activació que determina si es genera una sortida o no, i, si n'és el cas, genera una sortida que es transmet a altres nodes.

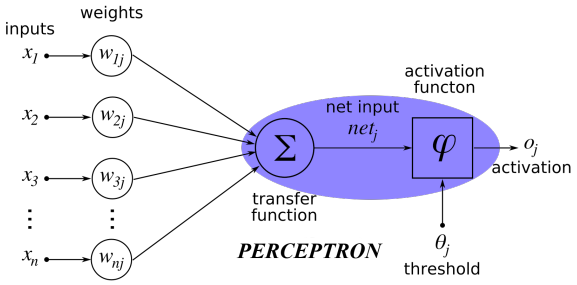

Figura 17. *Node d'una xarxa neuronal (perceptron*

L'estructura de les xarxes neuronals, com bé s'ha comentat, consisteix en un conjunt de nodes interconnectats. Aquestes interconnexions formen diferents capes en l'estructura de la xarxa:

- ***Input layer:*** capa d'entrada. És la capa que s'encarrega de rebre els paràmetres d'entrada a la xarxa neuronal.
- ***Hidden layers:*** capes ocultes. Es tracta d'una o múltiples capes intermèdies que s'encarreguen de realitzar les operacions pertinents a partir dels valors rebuts des de la capa d'entrada.
- ***Output layer:*** capa de sortida. A partir de les dades rebudes de les capes intermèdies, mostra els resultats obtinguts en aquestes.

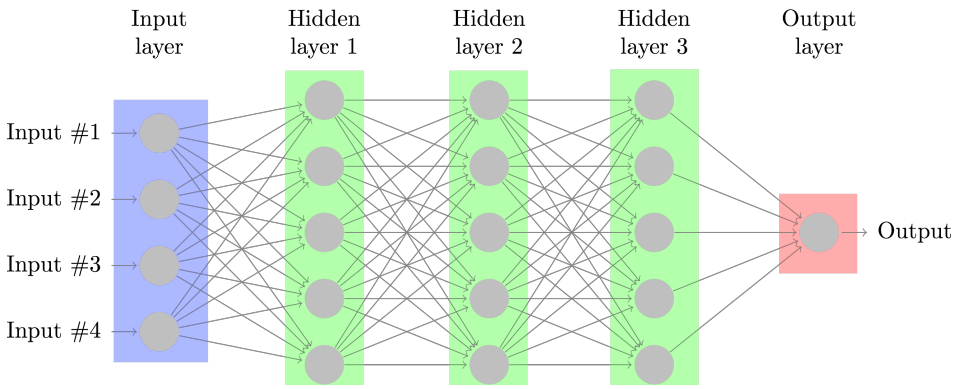

Figura 18. Estructura en capes d'una xarxa neuronal

5. Recerca i anàlisi

En aquesta secció, per una banda es pretén explicar els orígens, components i mercat de les smartglasses, i d'altra banda, es realitza un estudi en profunditat sobre la detecció d'objectes i les seves possibilitats.

5.1. Smart Glasses - estudi en profunditat

L'interès actual per a desenvolupar Wearables per part de moltes empreses del sector dels dispositius intel · ligents es materialitza en el mercat tecnològic de moltes formes i modalitats diferents. D'entre aquests, trobem les Smart Glasses, dispositius que afegeixen funcionalitats i informació a allò que l'usuari normalment reuniria del món real, apropant elements multimèdia a la visió del portador de forma còmoda i permetent prescindir de les mans durant la major part del temps que es consumeixi cert tipus de contingut. Qualsevol dispositiu ocular que compleixi aquestes característiques es considerat Smart Glasses. Un exemple en serien les ulleres que adapten el tint dels seus vidres a la intensitat de llum solar. En el nostre cas però, per la naturalesa de l'estudi que ens porta a realitzar aquest treball, només considerarem Smart Glasses aquells dispositius oculars computats, és a dir, digitals.

5.1.1. Història

En aquest punt es farà un repàs històric de l'evolució de les smartglasses al llarg de la historia i s'acabarà amb una visió actual del panorama aixi com el model seleccionat pel treball.

Al contrari del que es pot creure, les Smart Glasses digitals no son un invent sorgit del no-res durant aquesta ultima dècada. La realitat és què hi ha molts precursors a les ulleres actuals que daten a partir dels anys 60 [7] [8]:

[1968] *Sword of Damocles*: desenvolupat per Ivan Sutherland i Bob Sproull en l'Universitat de Utah va ser el primer dispositiu ocular digital del que es te constància. Concretament, era un dispositiu de Realitat Virtual que mostrava diferents wireframes de l'habitació depenent d'on mirava el portador.

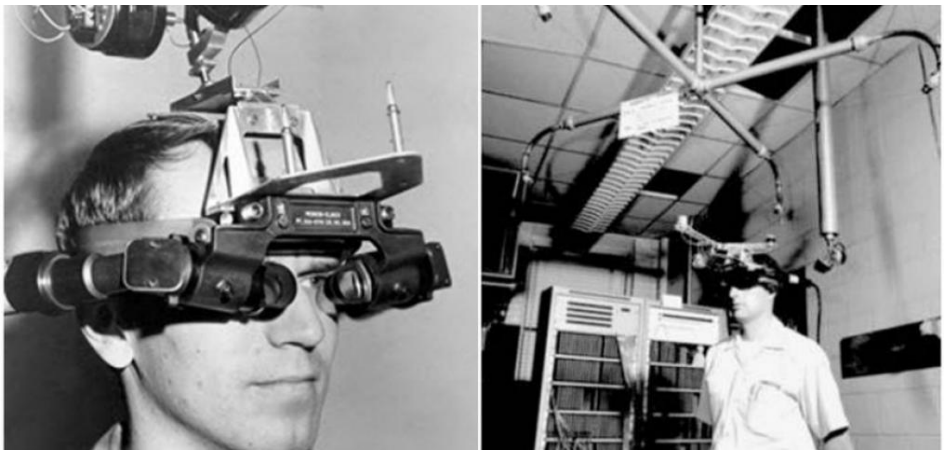

Figura 19. Sword of Damocles

[1980] *WearComp 1*: dispositiu que aplega diversos components informàtics per crear una experiència visual. Primer dispositiu desenvolupat per Steve Mann, considerat el pare dels dispositius oculars digitals i de la Realitat Mediada.

Figura 20. WearComp 1.

[1984] *WearComp 4*: evolució en la col · lecció WearComp que permet visualitzar imatges al ull esquerre i disposa de hardware addicional per la comunicació de dades, so i vídeo.

Figura 21. WearComp 4.

[1989] *The Private Eye*: desenvolupada per Steven Feiner, aquest dispositiu permet mitjançant tecnologia de reflexió escanejar un vector vertical de LEDs a traves del camp visual utilitzant un mirall vibratori. El resultat visual era poder visualitzar a una pantalla de 1,25 polzades objectes a una distancia de 18 polzades a una mida equivalent de 15 polzades.

Figura 22. The Private Eye.

[2000] *TASK-9*: dispositiu que incorpora per primer cop un ordinador a unes ulleres. Desenvolupades per l'empresa de Mark Spitzer, va crear moltes patents les quals van ser adquirides per Google quan l'empresa tanca al 2010.

Figura 23. Task 9.

[2005] *Nomad Digital Display*: encarades a àmbits d'automoció, proporcionen informació als conductors mitjançant una projecció laser a una pantalla sobre el seu camp de visió.

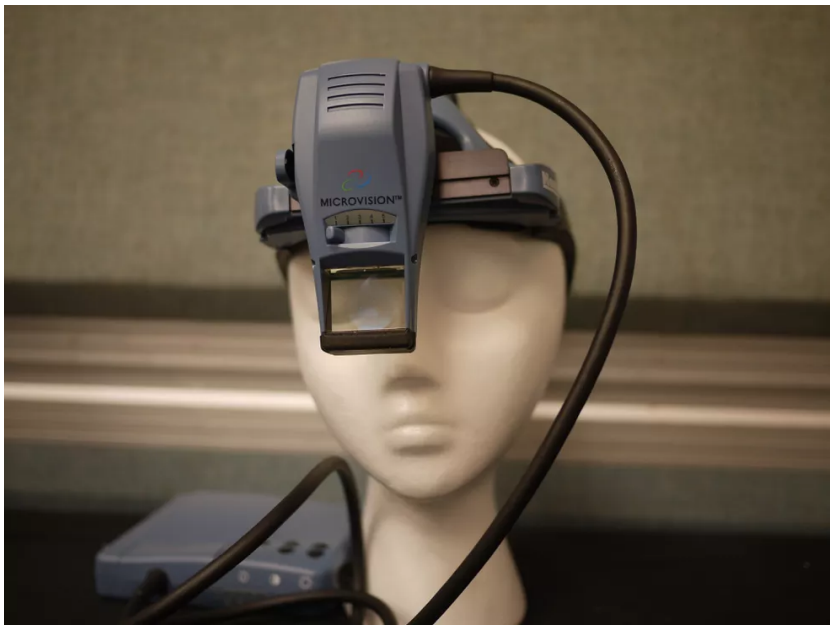

Figura 24. Nomad Digital Display.

[2008] *MyVu*: dispositiu que permet mitjançant un altre dispositiu de reproducció, reproduir imatge i so a traves de les ulleres.

Figura 25. MyVu.

[2011] *Wrap*: model de Vuzix de realitat virtual que compta amb un gran numero d'aplicacions, tant d'entreteniment com de recerca mèdica.

Figura 26. Wrap.

[2013] *Meta Prototype*: primer dispositiu composat de ulleres 3D estereoscòpiques i càmeres 3D creant un mon virtual en 3D que permet la interacció amb les nostres mans.

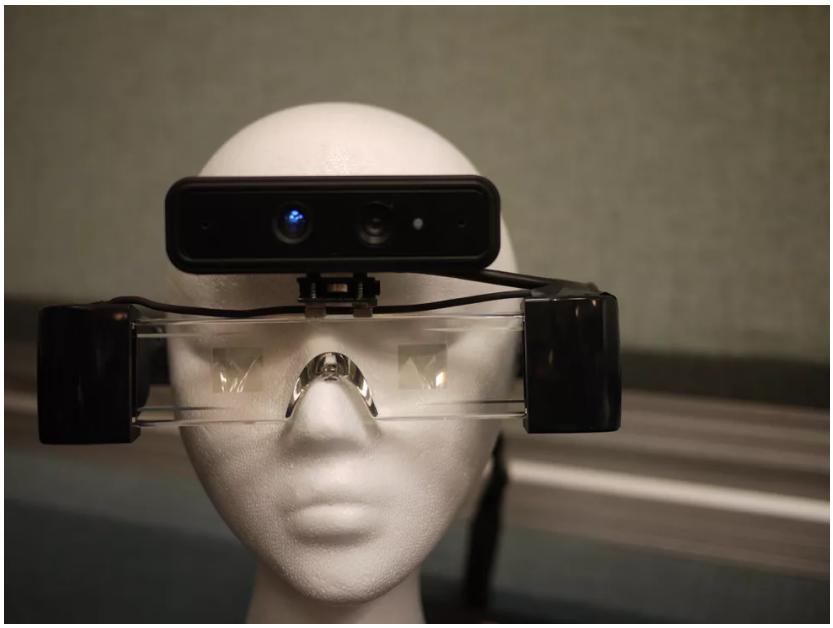

Figura 27. Meta Prototype.

[2013] *Google Glass*: primer dispositiu creat per Google que ofereix mans lliures per els smartphones, connexió a internet, comandes de veu i pantalla de realitat augmentada a tot color.

Figura 28. Google Glass.

Un cop vista la evolució fins a principis de la última dècada, empreses ja existents com Vuzix o Google s'han mantingut com líders en aquest sector, però els darrers anys han aparegut empreses amb una gran repercussió en el panorama dels dispositius de realitat virtual i realitat augmentada com Microsoft amb les seves "Hololens", Nreal o MagicLeap.

Si analitzem la situació, cap de les ulleres dels últims anys a revolucionat el mercat com passava a principis de segle, ja que gairebé cap dispositiu ha destacat per incorporar millores que despuntin d'entre la resta, només alguns que han incorporat la realitat virtual com a novetat en el mercat, com per exemple *Oculus*. Tot i això, segons les previsions de mercat actuals, aquest salt qualitatiu estarà marcat per altres factors [9] com: l'arribada de la connectivitat 5G, que reduirà la latència actual causada per xarxes 3G i 4G, o nous competidors en el sector com la possible arribada d'Apple.

El dispositiu que hem seleccionat és una de les opcions que proporciona l'empresa especialista en el sector *Vuzix*. Vuzix ha desenvolupat diversos models d'Smart Glasses i ulleres de Realitat Augmentada oferint solucions molt diverses per a empreses interessades en aquestes tecnologies. En el nostre cas hem seleccionat un dispositiu d'Smart Glasses desenvolupat sobre la plataforma Android, que disposa d'una pantalla que es situa al nivell de l'ull dret y incorpora una càmera disposada a la part exterior del dispositiu, permetent gravar el que es veu. Aquest dispositiu reuneix les característiques requerides per al desenvolupament de la solució que volem proposar, ja que, mitjançant la càmera del dispositiu, ens permetrà mantenir una imatge en viu del que l'usuari tindrà davant seu i sobre la qual disposarem la informació necessària en reconèixer els objectes visibles.

5.1.2. Components

Les smart glasses es tracten d'ordinadors adaptats de forma que puguin ser portats al cap. Hi han molts tipus de versions, i cadascuna té les seves particularitats específiques segons la funció desitjada. A continuació, s'esmentaran característiques que poden presentar aquests dispositius.

- La connectivitat sense fil. Aquesta pot ser de tipus Wifi, Bluetooth o totes dues alhora. Això permet, expandir les possibilitats que ofereix el propi dispositiu amb l'ús de la xarxa per accedir a Internet o la connexió amb un smartphone.
- El micròfon. Aquest element, present a la majoria de smart glasses permet la interacció de l'usuari mitjançant la veu. Al tractar-se de dispositius petits, aquesta forma de interacció facilita el seu ús gracies al assistent integrat en el dispositiu.
- La càmera. Segons el tipus de smart glasses aquestes requeriran de una o varies càmeres per captar el entorn i aplicar una capa virtual sobre aquest entorn.
- Projector. Permet visualitzar a la lent/pantalla la interfície visual que ens proporciona el dispositiu.
- Superfície tàctil i/o botons. Ofereix una alternativa a les comandes per veu si el usuari vol fer ús del dispositiu mitjançant els dits.

Figura 29. Components mostrats a un model especific de la marca Vuzix, les M300XL. Dispositiu amb el que el realitza el treball.

Entre d'altres, aquestes característiques són les més rellevants en termes generals ja que, existeixen altres com el cas del GPS però, ja estaríem parlant de especificacions segons el ús del dispositiu.

5.1.3. Funcionament de les Smart Glasses

El funcionament de les smart glasses és basa principalment en els elements que componsen el dispositiu, i per tant i han multitud de dispositius i molt variats entre ells. Això provoca una gran diversitat que comporta funcionaments molt diversos de un model a un altre. Seguidament analitzarem els principals punts comuns entre tots els models de

forma general.

En primer lloc tenim la part relacionada amb la visualització de la informació. Existeixen dos tipus de classificacions segons el numero de lents i segons el tipus de lents:

1. Número de lents

    - **Monocular**: disposa de una sola lent.
    - **Binocular**: disposa de dues lents augmentat el rang de visió.

2. Tipus de lents

    - **Curved mirror (o Curved combiner)**: el funcionament es basa en la llum emesa per una pantalla de cristall líquid (LCD) que arriba a un mirall còncau amb propietats de reflexió i transmissió que actua com a mirall/combinador, que reflexa la llum projectada a l'ull.

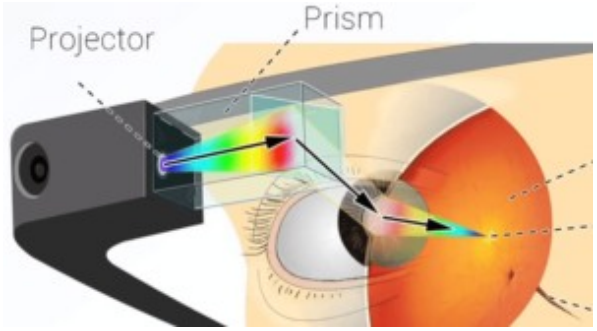

Figura 30. Esquema del funcionament de ulleres amb tipus de lents Curved mirror. [10]

    - **Waveguide (o Light-guide)**: mitjançant una pantalla de cristall líquid sobre silici (LCOS) que dispara els raigs de llum col · limats a través d'una reixeta de difracció que redirigeix la llum i eventualment forma una imatge expandida, que després es projecta en l'ull.

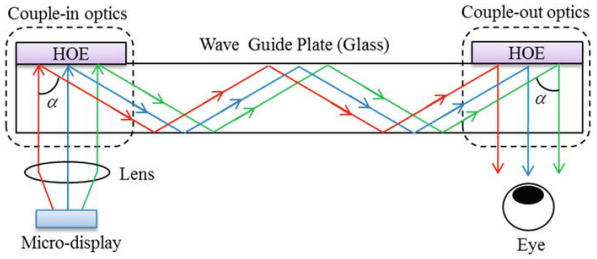

Figura 31. Esquema del funcionament de ulleres amb tipus de lents Waveguide.

En segon lloc, les ulleres presenten la capacitat de captar àudio i d'emetre'l. Si el dispositiu te a capacitat d'entrada d'àudio aquest serà més propens a acceptar comandes

de veu. De forma general, les ulleres utilitzen un mètode anomenat conducció òssia. Aquest mètode permet transmetre les vibracions d'àudio mitjançant la muntura de les ulleres a traves de un petit segment del crani. Les vibracions arriben a la oïda mitja on tres ossos transformen el so en representacions sonores significatives.

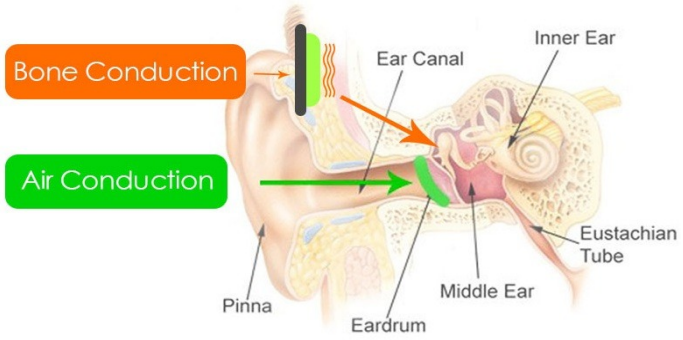

Figura 32. Esquema de la conducció òssia.

En tercer lloc, gracies a l'evolució de les càmeres, aquestes són avançades i de mida reduïda per ser incorporades sense cap inconvenient, permetent captar imatges/vídeos amb una qualitat de vídeo elevada.

Existeix un debat, explicat al apartat 5.1.4, sobre aquest component ja que, la privacitat de la persona es compromesa al no haver-hi cap indicador de si s'esta enregistrant o capturant imatges sense consentiment.

Finalment, respecte a la interacció amb les ulleres escollides [11], trobaríem dos estils d'interacció principals: interacció directa i llenguatge natural. La interacció directa seria definida per poder interactuar mitjançant el trackpad i els botons, en canvi, el llenguatge natural fora causat per poder interactuar amb les ulleres mitjançant ordres per veu gracies al micròfon incorporat. Es tracta d'un dispositiu Android que trasllada una interfície d'usuari similar a la d'un telèfon mòbil a la vista de l'usuari, per tant, la seva utilització corrent suposa una interacció de l'usuari amb un dispositiu aïllat, com passa en el paradigma de l'ordenador de sobretaula.

En realitzar la nostra aplicació per a aquest dispositiu, incorporem un altre concepte en la definició d'aquest, i és el paradigma de la realitat mixta. En el nostre cas, amb la detecció d'objectes plasmada en la realitat de l'usuari, ja que l'usuari només interactua amb el dispositiu centrant la seva vista i el sistema utilitza la informació de l'entorn com entrada implícita.

5.1.4. Privacitat

Com s'ha comentat al punt anterior, sorgeix un debat [12] dins d'alguns components de les ulleres. Per posar en context s'explicara mitjançant el cas de Google amb el seu model Google Glass.

La idea que algú pogués estar prenent fotos o filmant-te sense el teu coneixement (i consentiment) era incòmoda per a molts. Encara que les ulleres en realitat tenen una llum que indica quan es pren la foto o el vídeo, existia la preocupació que això pogués anul · lar-se amb un programari personalitzat.

El problema amb les filmacions / fotos sense consentiment en realitat no és un problema específic de les Google Glass, està relacionat amb els avanços tecnològics en general. Es poden demanar ulleres amb càmeres ocultes a un preu bastant econòmic i filmar d'una manera molt més discreta que amb Google Glass.

Va haver-hi preocupacions similars al reconeixement facial. En teoria, l'usuari de Glass podia caminar pel carrer i identificar a altres persones sobre la marxa. Combinant la gran quantitat d'informació que la gent pública a Internet i a les xarxes socials, és bastant fàcil obtenir una descripció general bastant completa d'un estrany basat en aquesta informació. Google va respondre a la preocupació i va prohibir les aplicacions de reconeixement facial. No obstant això, aquesta prohibició va ser més simbòlica que pràctica. En realitat, no és tan difícil carregar programari personalitzat (no controlat) en les ulleres.

Al projecte Glass, Google va enfrontar molts problemes i preocupacions de privacitat fins i tot abans del llançament del producte. La conseqüència porto a moltes empreses i establiments a prohibir l'ús d'ulleres intel · ligents.

Arrel del cas Google, moltes empreses han pres consciencia i s'han adaptat, però, això no vol dir que la violació de la privacitat no pugui passar en un futur proper o ja estigui passant.

5.1.5. Mercat

Amb el plantejament del treball, vam realitzar un estudi[5] exhaustiu sobre les smartglasses presents al mercat per tal d'adquirir un model i posar en pràctica el que s'esta exposant al document.

[5]Els preus mostrats a l'estudi s'obtenen entre els mesos de novembre i desembre de 2019. El preu fixat pot variar amb el valor actual dels dispositius.

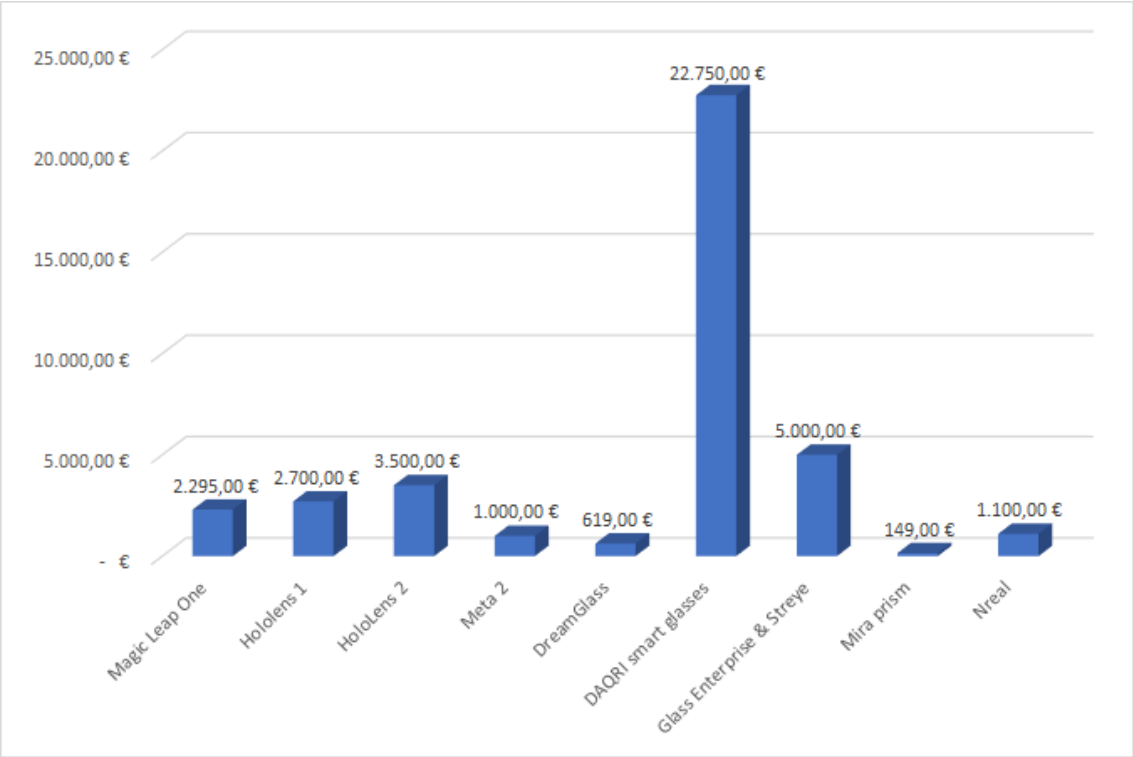

Figura 33. Mercat de smartglasses (desembre 2019).

Com es pot observar a la Figura 33, els preus generalment oscil · lent entre els 1000 € i els 5000 € amb alguna excepció que va més enllà com és el cas de les DAQRI. Aquest fet es pot explicar ja que la fabricació de pantalles hologràfiques per smartglasses binoculars tenen un gran preu de producció[6]. Analitzant també aquelles que estan per sota del llindar dels 1000€, trobem generalment smartglasses del tipus monoculars, ja que, el cost del tipus de lent és molt més baix i no és complex d'implantar.

El problema que s'ha trobat dins del mercat és la poca disponibilitat o no disponibilitat de la majoria dels dispositius, ja que la gran part, en ser noves creacions es basen en reserves les quals, ja estaven completes o nomes eren destinades a empresa. Per exemple, en el cas de les Nreal es va sol · licitar per correu i per la pàgina web poder accedir a la compra del dispositiu però, aquestes estaven esgotades o tenien llista d'espera d'un any. El mateix cas va ocórrer amb les MagicLeap, però, d'aquesta no vam rebre cap resposta directament.

Destacar que la gran majoria d'empreses noves en el sector creixen ràpidament i presenten un apartat per desenvolupadors, i algunes aporten un kit inicial per facilitar la tasca d'aprenentatge de desenvolupament sobre el dispositiu.

[6]Basats en fets històrics de la tecnologia, demostren que el preu anirà a la baixa, però fins que no passi un llindar acceptable no podem considerar que les smartglasses són acceptades per públic general.

5.2. Detecció d'Objectes

Ens trobem davant d'un sistema de Detecció d'Objectes quant aquest és capaç de, donada una imatge d'entrada, identificar de forma més o menys precisa quins objectes hi ha en aquesta. A més, qualsevol sistema de detecció d'objectes pot ser emprat per un altre sistema amb l'objectiu de prendre accions en raó al resultat obtingut. Per exemple, un cotxe autònom pot fer ús d'un sistema de detecció d'objectes per a detectar gossos amb l'objectiu de frenar o desviar el cotxe en detectar-los. Així doncs, el sistema de detecció ha de poder determinar amb precisió el posicionament de l'objecte. Per fer això, tot sistema de detecció d'objectes crea sobre la imatge un quadre delimitador entorn de l'objecte d'interès.

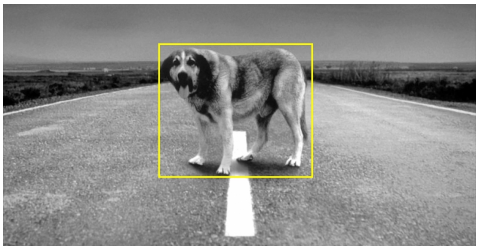

Figura 34. Exemple de detecció completa

En definitiva, qualsevol sistema de detecció d'objectes ha de complir dues condicions essencials per al correcte funcionament i aplicació:

- Identificar tots els objectes que són presents a la imatge i on estan situats.
- Filtrar l'objecte o tipus d'objectes als quals volem centrar l'atenció.

5.2.1. Mitjans per a la solució d'un problema de detecció d'objectes

Existeixen fins a 5 formes diferents d'afrontar un problema de detecció d'objectes. Les solucions existents se centren en diferents formes de tractar la imatge d'entrada, des del tractament més senzill i ineficient, fins al més complex i eficaç [13].

Solució 1: Divideix i conquereix

La solució més fàcil que es pot realitzar és dividir la imatge en 4 parts iguals que corresponen a les cantonades de la imatge. A continuació, passar aquestes parts a un classificador d'imatges, que ens retornarà si la part de la imatge té o no l'objecte desitjat. La part de la imatge que contingui l'objecte serà delimitada completament, quedant la imatge de la següent forma:

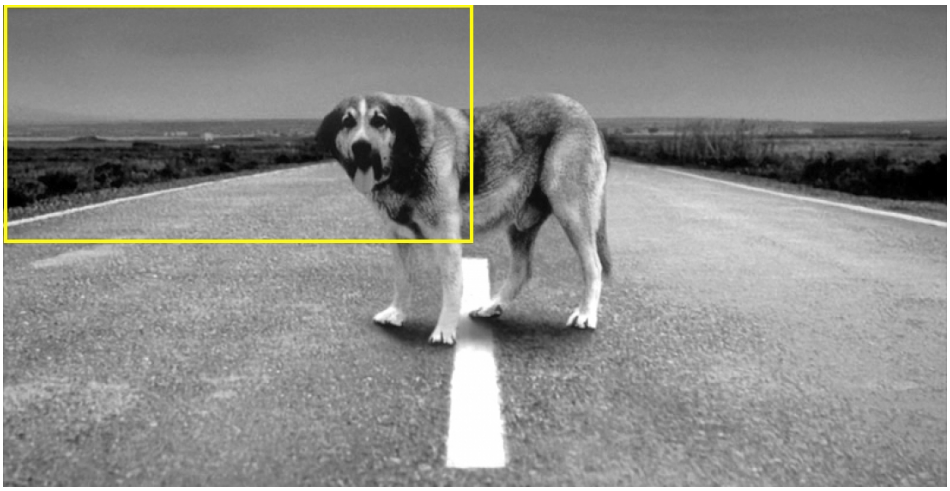

Figura 35. Solució 1: Divideix i conquereix

El problema d'aquesta solució és la poca precisió que proporciona, ja que només detecta les parts de l'objecte que siguin més identificatives d'aquest. Això pot comportar problemes greus en un sistema que necessita tenir l'objecte localitzat correctament.
En el cas de la Figura 35, per exemple, gran part del cos del gos no es té en compte i això pot suposar que el cotxe autònom comentat en la secció anterior prengui una decisió que perjudiqui el gos.

Solució 2: Increment del nombre de divisions

Per solucionar els problemes de la solució anterior, aquesta proposa fer moltes més divisions de la imatge de forma exponencial i no ordenada, generant una sortida com la següent:

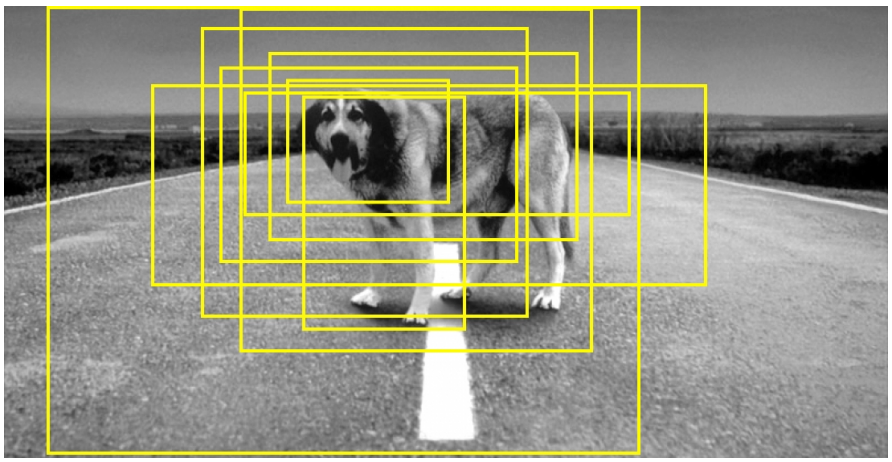

Figura 36. Solució 2: Incrementar les divisions

El problema d'aquesta solució és que, degut a la quantitat de quadres delimitadors existents que contenen el mateix objecte, és difícil de controlar la sortida i de prendre decisions en raó a aquesta.

Solució 3: Divisions estructurades

Per tal de millorar la solució 2, es proposa fer divisions més estructurades, seguint un patró determinat per tractar la imatge d'entrada, prèviament dividida en una matriu: per cada cel · la de la matriu es localitza el punt central d'aquesta i, a partir d'aquest punt, es prenen 3 seccions de diferents mides, les quals tenen aquest punt com a punt central. Aquestes tres seccions seran les que es passaran al classificador d'imatges. Un cop fet això, el nombre de divisions seran més acurades, reduint així el nombre de quadres delimitadors i obtenint una solució més precisa.

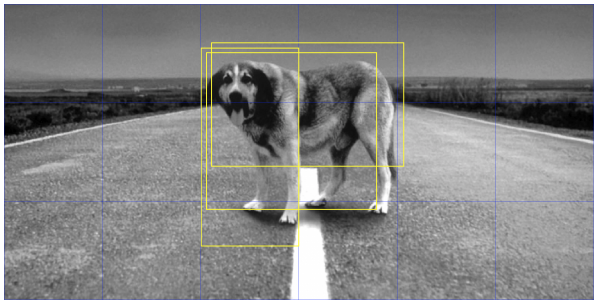

Figura 37. Solució 3: Divisions estructurades

Solució 4: Ser més eficients

La solució anterior es pot millorar augmentant el rang de la matriu, obtenint així major precisió per objectes que es troben a menys escala respecte a la imatge a la qual pertanyen, ja que, en disminuir la mida de les cel · les de la matriu, agafes seccions més focalitzades de la imatge.

Una altra solució és augmentar el nombre de seccions per cada punt central de les cel · les, agafant relacions d'aspecte més variades, el qual permetrà generar una varietat més àmplia de quadres delimitadors, entre els quals trobarem alguns que posicionin l'objecte de forma més precisa.

Fet tot això, el següent pas seria realitzar un pas intermedi que detecti els quadres delimitadors que indiquin el mateix objecte, escollir aquells que posicionin l'objecte més correctament i desestimar la resta.

Solució 5: Utilitzar Deep Learning

El ***Deep Learning*** és un camp del *Machine Learning* que s'enfoca en els algorismes que defineixen les estructures i el funcionament de les xarxes neuronals artificials.

Fent ús d'aquestes xarxes neuronals, es pot aconseguir una millor solució al problema de la detecció d'objectes:

- En lloc d'obtenir seccions de la imatge per calcular, es pot passar a la xarxa neuronal la imatge original per tal que aquesta en redueixi les dimensions de forma òptima.

- Es pot utilitzar una xarxa neuronal per a seleccionar les seccions de la imatge de forma selectiva.

- Es pot consolidar un algorisme de *Deep Learning* perquè doni prediccions tan acurades com siga possible al quadre delimitador original, fent que aquest sigui el més fi i estret possible.

El resultat obtingut a partir de la utilització de xarxes neuronals i *Deep Learning* ens generaria una sortida com la següent:

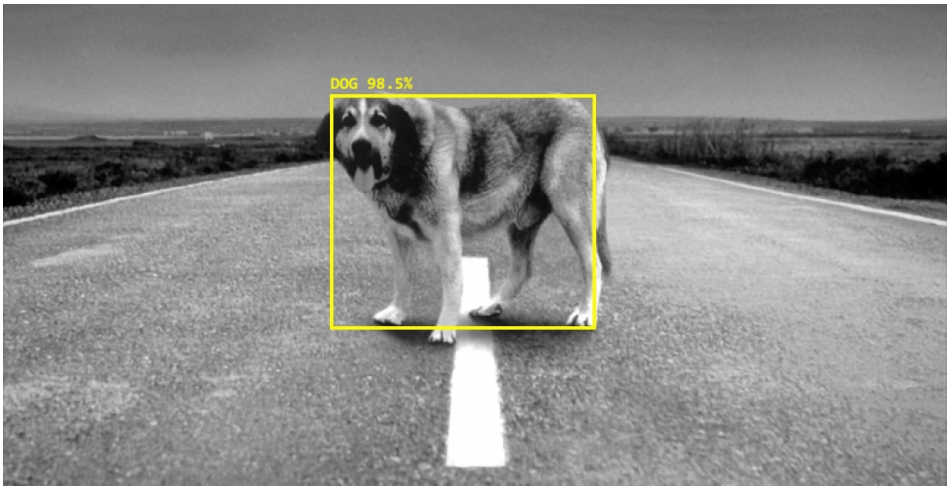

Figura 38. Solució 3: Divisions estructurades

Hi ha diferents tipus d'algorismes de *Deep Learning* que ens permeten realitzar la detecció i el reconeixement d'objectes per a una imatge, tots ells utilitzen com a base les xarxes neuronals de tipus **CNN (Convolutional Neural Network)**.

Les xarxes CNN són unes xarxes neuronals que processen una imatge d'entrada generant diferents capes d'aquesta mitjançant filtratges (*Convolutions*) i reduccions d'espai (*Poolings*) sobre els seus píxels per tal d'extreure característiques que permetin identificar els objectes desitjats.

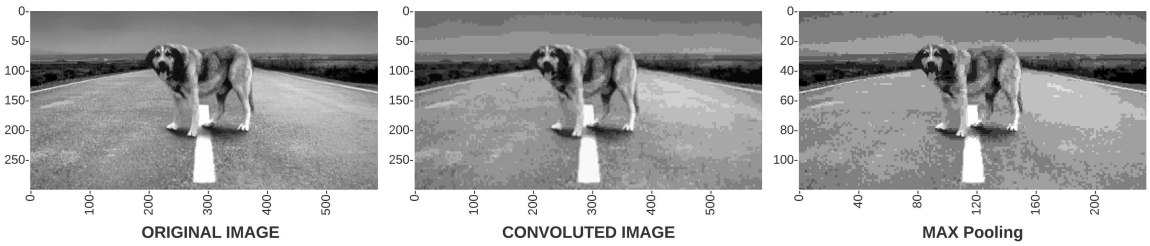

Figura 39. Imatge passada per les diferents capes

La xarxa neuronal completa es veu i funciona de la següent forma [14]:

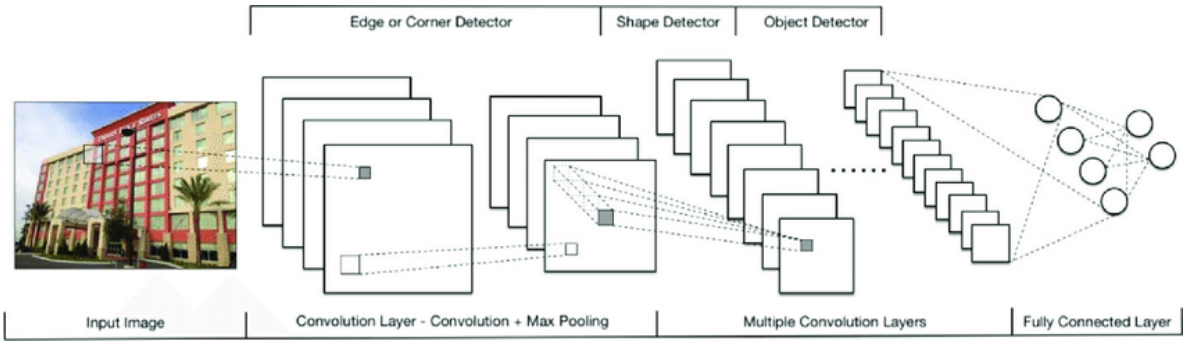

Figura 40. Esquema d'una xarxa CNN

1. Passem una imatge d'entrada a la primera capa de convolució, la qual, aplicant diversos filtratges diferents, extraurà característiques molt genèriques de la imatge. El filtratge consisteix a operar un cert conjunt de píxels amb uns valors determinats anomenats pesos. El resultat d'aquestes operacions fa que en la imatge resultant els píxels de les característiques desitjades destaquin sobre la resta de píxels.

2. Cada un dels filtres hauria de proporcionar característiques diferents de la imatge. Per mantenir la mida de la imatge i no perdre informació dels extrems d'aquesta, en el procés d'extracció de característiques, s'utilitza un contorn de píxels a zero. No fer-ho ajuda a reduir el nombre de característiques (pel precís fet que es perd informació dels extrems), però es redueix (en poca mesura) la imatge.

3. Si es desitja fer una reducció dels paràmetres més radical, es realitza un *Pooling*, que, com s'ha comentat anteriorment, s'encarrega de les reduccions d'espai de la imatge.

4. Abans de fer la predicció es realitzen múltiples passades per les capes de convolució i *pooling*. Com més profunditat s'aconsegueixi en la xarxa neuronal, és a dir, com més iteracions, més especifiques seran les característiques extretes de la imatge.

5. La capa de sortida de la xarxa CNN és una capa totalment connectada, on l'entrada provinent de les altres capes es converteix en una seqüència d'una dimensió (una seqüència de píxels) que conté totes les característiques. Aquesta s'envia posteriorment a un classificador que ho transforma en el nombre de classes que desitgi la xarxa.

6. La sortida actual es compara amb la sortida anterior de la xarxa CNN i, mitjançant una funció definida a la capa de sortida de la xarxa, es calcula un gradient d'error.

7. L'error es propaga cap a capes superiors per tal d'actualitzar els filtres (els pesos de càlcul).

8. Un cicle d'entrenament de la xarxa neuronal es completa un cop s'ha fet una passada endavant i la seva corresponent passada endarrere.

Per si mateixes, aquestes xarxes neuronals calculen moltes regions d'una mateixa imatge, suposant un alt cost computacional. Per solucionar això, s'han realitzat diferents algorismes que, partint d'una xarxa CNN, aconsegueixen augmentar l'eficiència d'aquesta xarxa neuronal [15, 16].

R-CNN

L'algorisme R-CNN (*region-based CNN*) redueix el nombre de regions calculades proposant un conjunt determinat de regions en la imatge en les quals es comprovarà l'existència d'objectes identificables. Per escollir aquestes regions l'algorisme utilitza cerca selectiva.

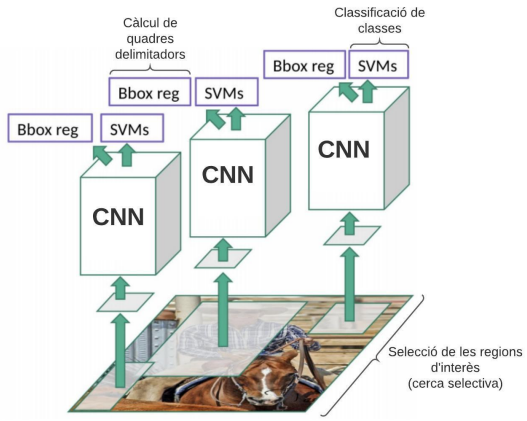

Figura 41. RCNN

La cerca selectiva consisteix a dividir la imatge en múltiples regions, de les quals, les similars es combinaran segons 4 patrons diferents (escales, colors, textures i recintes) per formar regions més grans, o regions d'interès, sobre les quals es farà la detecció d'objectes en la xarxa CNN.

Així doncs, la detecció es realitza de la mateixa forma que hem vist anteriorment en l'explicació de les xarxes CNN, però aquest cop l'entrada a la xarxa és cada una d'aquestes regions d'interès, les quals són modificades per tenir la mida necessària per a l'entrada a la xarxa. Un cop la xarxa CNN ha extret les característiques de cada regió, aquestes regions es classifiquen segons la classe de la qual formen part i es calcula el quadre delimitador de la regió identificada.

Tot i que, dividint la imatge en regions d'interès, ens evita que la xarxa CNN busqui característiques en llocs de la imatge on clarament no n'hi ha, aquest tipus d'algorisme segueix tenint problemes:

- Entrenar la xarxa segueix requerint molt temps a causa de haver de classificar 2000 propostes de regió per imatge (2000 passades per la xarxa CNN).
- No es pot aplicar a la vida real, ja que tarda 47 segons per cada imatge.
- La part de cerca selectiva és un algorisme estàtic sense aprenentatge, la qual cosa pot derivar en propostes de regions errònies.

Fast R-CNN

Per solucionar el problema de les 2000 execucions de R-CNN, l'algorisme Fast R-CNN proposa executar la xarxa CNN un sol cop per tota la imatge i abans de passar els resultats a l'última capa (la capa totalment connectada que genera la seqüència d'una dimensió), fer-n'hi en aquest punt la cerca selectiva de regions, les quals seran adaptades en mida i enviades a aquesta última capa de forma individual; posteriorment, seran processades per a la classificació de classes i generació de quadres delimitadors.

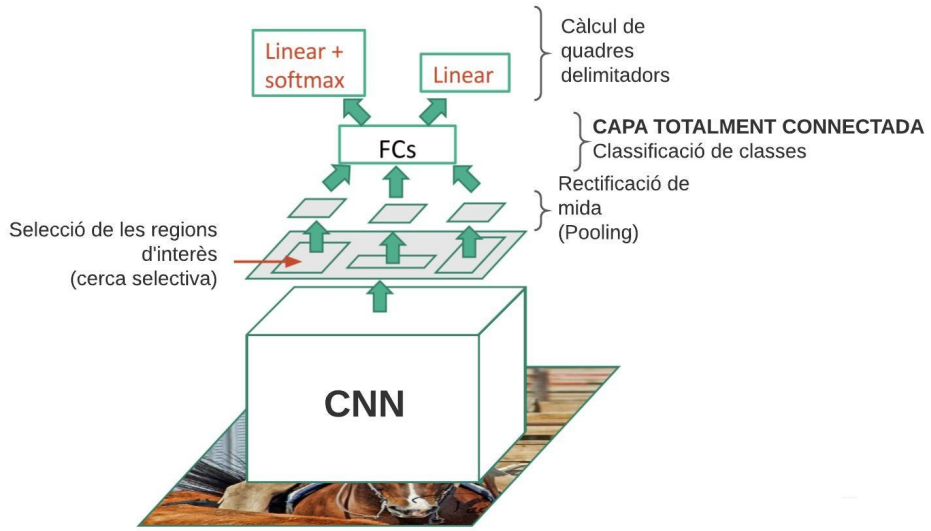

Figura 42. Fast RCNN

Arribats a aquest punt, l'únic problema que presenta l'algorisme Fast R-CNN després de solucionar el problema del temps d'execució de la xarxa CNN, és la pèrdua de rendiment en fer les propostes de regions d'interès. Aquest problema és solucionat per l'algorisme que veurem a continuació.

Faster R-CNN

L'algorisme Faster R-CNN consisteix en un algorisme Fast R-CNN el qual substitueix la cerca selectiva per una xarxa neuronal que s'encarrega de generar les propostes de regions sobre la imatge. Aquesta xarxa s'anomena *Region Proposal Network* (RPN).

La xarxa RPN consisteix en un algorisme de finestra lliscant, on cada finestra genera k caixes de diferents mides. Per cada caixa es fa la predicció de dues coses:

- Probabilitat de què la caixa contingui un objecte.
- Valor del quadre delimitador per ajustar millor les caixes.

Així doncs, Faster R-CNN aplica la xarxa RPN a la sortida de la xarxa CNN i retorna les propostes d'objectes amb la seva puntuació. Aquestes propostes posteriorment seran

adaptades i plasmades un altre cop a la sortida de la xarxa CNN abans de passar pels classificadors i generadors de quadres delimitadors.

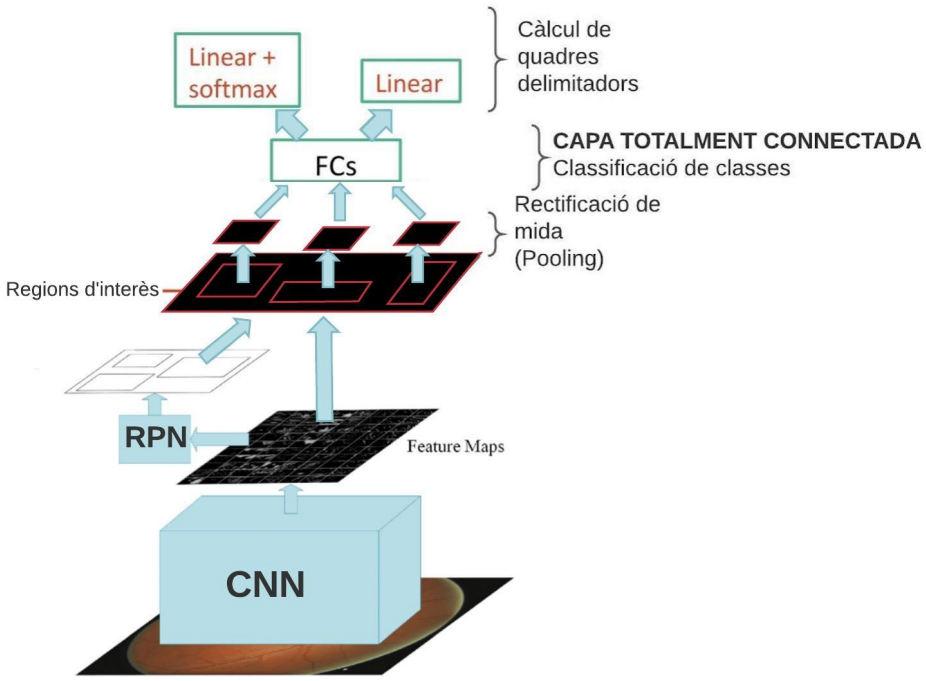

Figura 43. Faster RCNN

Els problemes amb els quals ens podem trobar arribats a aquest punt tenen relació amb l'extracció dels objectes, i és què per poder detectar quines regions es poden extreure segons similitud i extreure tots els objectes existents, cal recórrer diverses vegades tota la imatge. A més, hi ha diferents sistemes (CNN, RPN, Capa Totalment Connectada) treballant que depenen l'un de l'altre i que, per tant, el seu funcionament depèn de com ha treballat el sistema anterior. Així doncs, si un presenta errors en els seus càlculs, aquests es transmeten a la resta de sistemes; o si un és menys eficient, juntament amb la resta formen un sistema general poc eficient.

YOLO (*You Only Look Once*)

A diferència de la resta d'algorismes que hem vist, que fan diverses iteracions sobre la imatge per poder generar les regions amb més possibilitat de contenir un objecte, les anomenades regions d'interès, YOLO proposa una forma més eficient de processar una imatge. YOLO, en lloc d'iterar en cerca de regions amb objectes dintre, fa divisions controlades de la imatge en una sola execució [17].

Figura 44. Divisió de la imatge en una matriu d' $S\ x\ S$

En aquest algorisme, la imatge d'entrada a la xarxa CNN es divideix en una matriu d' $S\ x\ S$ cel · les. Cada cel · la prediu B quadres delimitadors, així com C probabilitats de classe.
La predicció del quadre delimitador té *5* components: (x, y, w, h, confiança):

- Les coordenades (x, y) representen el centre del quadre, en relació amb la ubicació de la cel · la de la quadrícula (recordar que, si el centre del quadre delimitador no cau dins de la cel · la de la quadrícula, aquesta cel · la no se'n fa responsable). Aquestes coordenades es normalitzen entre 0 i 1.

- (w, h) representen les dimensions de la caixa en relació amb la mida de la imatge. També es normalitzen entre 0 i 1.

- La confiança reflecteix la presència o absència d'un objecte de qualsevol classe.

De forma paral · lela, cada cel · la genera les seves probabilitats de classes d'entre les classes que contempla la base de dades.

(a) Quadres delimitadors **(b)** Probabilitats de classes

Figura 45. Prediccions fetes en la xarxa CNN

En aquest punt, les prediccions de classe i el resultat de confiança dels quadres delimitadors s'uneixen en un resultat final que ens indica quant segur n'està l'algorisme de què en un quadre delimitador determinat hi ha un objecte d'una classe específica.

Finalment, fet que el pas anterior genera prediccions per a un nombre molt elevat de quadres delimitadors, a partir d'un valor mínim de confiança s'eliminen aquells quadres delimitadors que tenen la seva confiança per sota d'aquest mínim.

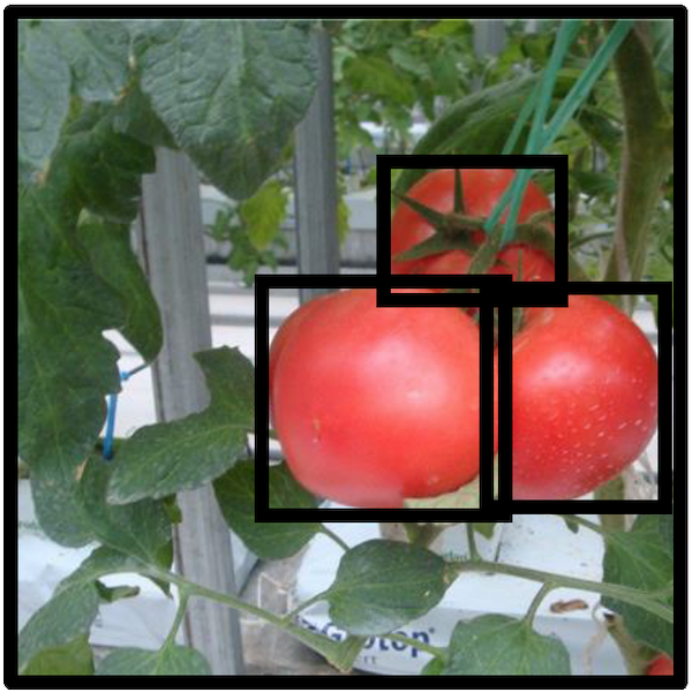

Figura 46. Resultat final de la unió de les dues prediccions

Així doncs, YOLO proposa una forma de divisió de la imatge que, a diferència de la finestra lliscant, no requereix d'iteracions sobre la imatge i que, a més, comparat amb Faster R-CNN, fa menys de la meitat d'errors en el fons de la imatge gràcies al fet de controlar millor el context del global d'aquesta.

Tot i això, l'algorisme tenia una limitació en les primeres versions, i era la dificultat per detectar objectes petits. Això era degut als valors espacials de l'algorisme, i és que les cel · les en què es dividia la imatge podien no ser prou precises en píxels, perdent així informació d'objectes petits. YOLOv3 soluciona aquests problemes generant tres capes diferents amb diferents resolucions de la mateixa imatge, una per a objectes grans, una altra per a objectes mitjans i una última per a objectes petits, de tal forma que les cel · les en les diferents capes són més o menys acurades en la identificació dels píxels.

6. Desenvolupament de l'eina de Detecció d'Objectes per a l'Awareness

A continuació s'explicarà el perquè de les tecnologies emprades per a la realització de l'aplicació, mitjançant un breu estudi d'aquestes; i quin ha estat el desenvolupament de l'aplicació, mostrant els punts més importants del codi implementat.

6.1. Tecnologies emprades per a la detecció d'objectes

Per a realitzar l'aplicació que es presenta en aquest treball s'ha optat per utilitzar unes tecnologies en concret per a cada punt clau d'aquesta:

1. **Smart glasses de Vuzix:** dispositiu.
2. **OpenCV:** per tal d'aportar tot l'entorn de detecció d'objectes a l'aplicació.
3. **COCO Dataset:** base de dades classificadora d'imatges.
4. **Darknet:** implementació *open source* d'una xarxa neuronal.
5. **YOLOv3:** algorisme que aplicarem a la xarxa neuronal per a fer la detecció d'objectes.

A continuació, s'explicara perquè s'han escollit aquestes tecnologies y no d'altres igual de competitives.

6.1.1. Smart Glasses

Per la realització d'aquest projecte, s'ha requerit de la utilització de dos dispositius diferents. El primer d'ells es tracta del model de ulleres de realitat mixta de Vuzix, les M300XL tal i com es mostra a la Figura 47. Respecte les especificacions[7] més rellevants, tenim un dispositiu amb les següents característiques:

- Dual Core Intel Atom CPU
- 2GB system RAM
- OS Android 6

Figura 47. Ulleres de realitat mixta de la marca Vuzix model M300XL.

[7]Per a més informació sobre les especificacions detallades del dispositiu anar a https://www.vuzix.com/products/m300xl-smart-glasses#/m300xl-technical-specs

Després de realitzar la integració bàsica i els primers tests, es va trobar que la capacitat computacional d'aquest model no era suficient per poder executar l'aplicació amb els fotogrames per segon mínims. L'alt nombre de còmputs a realitzar per l'aplicació en detectar objectes repercuteix en el rendiment a causa de falta de capacitat computacional. Aquest anàlisi s'obté mitjançant el calcul de frames per segon on, amb el model M300XL, s'arriba a una mitja de 0,05 FPS (frames per segon). Per poder realitzar aquests càlculs satisfactòriament amb la velocitat adequada per poder oferir els frames per segon mínims, es requereix d'una solució: hardware amb més memòria RAM i CPU més potent.

El segon dispositiu es tracta del model de ulleres de realitat mixta desenvolupat per Vuzix, les M400 tal i com es mostra a la Figura 48. Amb aquestes es va poder solucionar el problema de rendiment que hi havia amb les anteriors. Es tracata d'un dispositiu amb les següents característiques:

- 8 Core 2.52Ghz Qualcomm XR1
- 6GB LPDDR4 RAM
- OS Android 8.1

Figura 48. Ulleres de realitat mixta de la marca Vuzix model M400.

Aquest segon dispositiu ofereix la solució al problema exposat al primer dispositiu. Aquest dispositiu de Vuzix ofereix una CPU dedicada a dispositius mòbils amb una freqüència de rellotge[8] de 2.53 GHz, molt més elevat que el model anterior el qual només presentava 1.8 GHz. Analitzant la memòria RAM, aquest segon dispositiu presenta una memòria més gran i més ràpida que el seu model predecessor. Amb aquest conjunt de factors principals trobem la solució de potència que mancava amb el dispositiu anterior i l'aplicació pot funcionar amb un rendiment estable. Aquí es torna fer l'anàlisi pertinent i s'obté una mitja de més de 30 frames per segon, per sobre de la mitja que l'ull humà detecta com a moviment [18]. Cal comentar que el sistema operatiu, tot i no ser el principal causant de major rendiment és un factor a tenir en compte, ja que, proporciona millora del software base on l'aplicació s'ha d'executar.

[8]Velocitat a la qual un ordinador realitza les seves operacions més bàsiques.

Respecte a l'elecció d'aquest tipus d'ulleres, principalment ha estat condicionada pel mercat, en el qual, les solucions disponibles interessants per al nostre objectiu eren poques. D'entre les solucions possibles, la majoria, eren molt cares o no estaven disponibles per a la compra al públic. A més, moltes d'elles tenien un software més complicat per poder desenvolupar-hi una aplicació.

Com que el nostre objectiu era poder mostrar una detecció d'objectes en l'entorn actual de l'usuari, es requeria fonamentalment una càmera per a la lectura de l'entorn i una superfície on mostrar la detecció. Així doncs, el dispositiu seleccionat, que compleix amb aquests dos requeriments i que, a més, inclou un sistema ja conegut per nosaltres com és Android, es va convertir en l'opció més adequada.

6.1.2. OpenCV *(Open Source Computer Vision Library)*

Llibreria de software de visió per computador i machine learning de codi lliure desenvolupat inicialment per Intel el 1998 i publicat per primer cop l'any 2000 amb l'objectiu de proporcionar una infraestructura per aplicacions de visió per computadors [19] [20]. Considerada com la llibreria més popular[9] gràcies a característiques com:

- Llicencia BSD *(Berkeley Software Distribution)* que permet utilitzar la llibreria de forma lliure, sigui per propòsits comercials o d'investigació.
- Multiplataforma. Present en sistemes operatius GNU/Linux, Windows, Mac OS X i Android.
- Documentada i explicada amb una gran comunitat de desenvolupadors on compartir articles i tutorials, tant per iniciats o experimentats en visió per computadors.
- Desenvolupat en C++ i una API que permet fer ús de la llibreria amb llenguatges com Python, Java, Matlab, Octave i Javascript.

La llibreria compta amb més de 2500 algoritmes optimitzats, entre ells, nous algoritmes en l'estat d'art actual de la visió per computadors i machine learning. Aquesta varietat permet que la llibreria tingui aplicacions en diferents àmbits com reconeixement d'objectes, robòtica mòbil, realitat augmentada, etc.

S'ha escollit aquesta llibreria per al desenvolupament del treball degut a les grans possibilitats que ofereix en el camp de la visió per computadors. Ofereix moltes eines per al processament d'imatges i de la càmera del dispositiu, així com un fàcil medi d'utilització d'eines de detecció d'objectes com seria la xarxa neuronal Darknet; tot plegat, suposa la part més important de l'aplicació realitzada, i OpenCV ens facilita l'accés a aquests recursos (vegeu la seva utilització pràctica en l'apartat 6.2).

[9] Estudi realitzat per Svitla el novembre de 2019, publicat a la pàgina web: `https://svitla.com/blog/overview-of-modern-computer-vision-tools`

6.1.3. Darknet i YOLO

Darknet és un framework open source de Xarxes Neuronals sorgit en el 2014. El framework està escrit en C i CUDA la qual cosa el converteix en un framework ràpid i fàcil d'instal · lar.

Treballant juntament amb YOLOv3, algorisme explicat en l'apartat **5.2.1 - YOLO (*You Only Look Once*)**, se situen al capdavant en rapidesa i precisió gràcies al codi en C, que permet un gran rendiment en execucions en la CPU. A més, es pot aconseguir moltíssima més velocitat en executar el codi en la GPU, permès mitjançant el codi escrit en CUDA.

L'elecció d'aquesta xarxa neuronal ha estat condicionada per la utilització de la llibreria OpenCV, la qual incorpora eines d'integració que permeten llegir algorismes implementats en aquest tipus de xarxes neuronals. L'algorisme per el qual aquesta xarxa neuronal es tant reconeguda, es YOLO, per la qual cosa, la utilització d'aquest ha estat imperativa.

6.1.4. COCO Dataset

Segons es defineixen ells mateixos en la seva pàgina web[10]:

"*COCO és un conjunt de dades de detecció, segmentació i subtitulació d'objectes a gran escala*"

Algunes de les seves característiques més destacables són:

- Reconeixement en context.
- 330 mil imatges amb més de 200 mil imatges etiquetades.
- 1.5 milions d'objectes instanciats.
- 80 categories d'objectes.
- 91 categories addicionals per a agrupacions d'objectes.
- 5 captures per cada imatge.
- 250.000 persones y els seus punts claus.

Degut a les seves destacables característiques, la xarxa neuronal Darknet utilitza COCO com a base de dades per a l'entrenament dels seus algorismes. Així doncs, degut a la utilització inicial d'un model de l'algorisme YOLO entrenat per Darknet, necessitem de la utilització de dades obtingudes de COCO, per tal de fer la detecció de forma correcta amb els models dels que disposem.

[10] `https://cocodataset.org`

6.2. Implementació

Com bé s'ha comentat amb anterioritat, l'objectiu principal del treball és desenvolupar una eina de detecció d'objectes en un dispositiu de smartglasses, per tal de disposar d'un entorn propi amb aquestes tecnologies que puguem adaptar a un context d'awareness de lliure elecció.

Així doncs, l'eina desenvolupada consisteix en una aplicació android executable en la gamma *M*[11] de smartglasses de *Vuzix*. L'elecció d'aquest tipus d'aplicació ha estat condicionada per les mateixes smartglasses, les quals són dispositius android[12]. Tot plegat s'ha desenvolupat en l'entorn que el sistema android ofereix per a desenvolupadors: Android Studio[13].

En l'aplicació s'hi poden distingir 4 seccions principals que tenen gran importància en el correcte funcionament d'aquesta i que seran posteriorment vistes en detall en les seves respectives seccions:

1. ***Firebase*:** s'encarrega de l'emmagatzematge al núvol dels fitxers necessaris per a realitzar la detecció d'objectes.
2. **Gestió de sensors:** control de diferents canvis ambientals en l'entorn de l'usuari per tal d'adaptar la detecció d'objectes a aquests.
3. **Control i gestió de la càmera:** se centra en la total disposició de la càmera del dispositiu per tal de poder detectar les imatges en viu de l'entorn de l'usuari.
4. **Detecció d'objectes:** secció de l'aplicació on es processa la imatge obtinguda de la càmera per a la detecció dels objectes existents en l'entorn de l'usuari.

6.2.1. *Firebase* - emmagatzematge al núvol

Per tal de poder detectar objectes, hem de configurar la xarxa neuronal amb els fitxers corresponents: configuració de com ha de funcionar la xarxa (*yolov3.cfg*) i model de pesos amb l'entrenament previ de l'algorisme (*yolov3.weights*). També necessitem obtenir una llista amb les classes de la base de dades que fa servir la xarxa neuronal, en aquest cas, per Darknet, necessitem la base de dades COCO i el seu fitxer *coco.names*.
Aquests fitxers tenen un elevat pes a causa de la gran informació que incorporen, per la qual cosa, no és possible carregar aquests fitxers amb l'aplicació, així doncs, cal carregar-los en l'emmagatzematge al núvol i descarregar-los posteriorment.

El codi necessari per poder treballar amb *Firebase* és el següent:

```
1    // camp en la definició de la classe
2    private StorageReference mStorageRef;
```

[11] En aquest treball s'han utilitzat els models M300XL i M400, comentats en apartats anteriors.
[12] Les M300XL disposaven de la versió 6 (Marshmallow) i les M400 de la versió 8.1 (Oreo).
[13] https://developer.android.com/studio.

```
    protected void onCreate(Bundle savedInstanceState){
        mStorageRef = FirebaseStorage.getInstance().getReference();
    }
```

```
    private void download() {
        StorageReference namesRef = this.mStorageRef.child("file_name");

        File localFile;
        localFile = new File(getApplicationInfo().dataDir + File.separator
                                + "file_name");

        File finalLocalFile = localFile;
        namesRef.getFile(localFile)
                .addOnSuccessListener(taskSnapshot -> {
                    Log.d("NAMES_DOWNLOAD", "File downloaded");
                    FileInputStream namesStream = null;
                    try {
                        namesStream = new FileInputStream(finalLocalFile);
                    } catch (FileNotFoundException e) {
                        e.printStackTrace();
                    }
                })
                .addOnFailureListener(exception ->
                    Log.d("NAMES_DOWNLOAD", "File not downloaded"));
    }
```

El codi anterior serveix per descarregar qualsevol fitxer prèviament carregat a través de la consola web de l'emmagatzematge de *Firebase*.

6.2.2. Gestió de sensors

Una de les accions que s'ha dut a terme per tal de millorar el rendiment de l'aplicació ha estat filtrar els frames que s'han de detectar i els que no.

En aquest cas, s'utilitza l'acceleròmetre per a identificar quan l'usuari està realitzant un moviment amb el cap per així no fer-hi deteccions, ja que aquestes no anirien en consonància amb el moviment.
Així doncs, en una activitat asíncrona a l'activitat principal es fa la lectura de l'acceleròmetre, actualitzant per cada detecció un booleà que indica si hi ha hagut un moviment elevat o no. Aquest booleà serà accedit per l'activitat principal en cada frame, per tal de decidir si ha de fer deteccions pel frame actual, en cas de no haver-hi un moviment elevat, o no fer-n'hi, en cas d'haver-hi moviment.

6.2.3. Llibreria OpenCV

Un dels requeriments que van sorgir arran de la col · laboració amb *Invelon* va ser l'estudi i utilització de la llibrería OpenCV com a llibreria facilitadora de les eines de visió per a computadors.

Aquesta llibreria ens aporta el control i lectura de la càmera del dispositiu i eines per al tractament i processament d'imatges. En relació amb aquest últim fet, trobem eines per a gestions tals com la detecció d'objectes.

Per tal de poder treballar amb la llibreria OpenCV primer cal carregar-la en inicialitzar l'aplicació (funció *onCreate()*) i comprovar que s'ha carregat de forma correcta:

```
1    protected void onCreate(Bundle savedInstanceState){
2        if (!OpenCVLoader.initDebug()) {
3            Toast.makeText(getApplicationContext(), "There's a problem",
4                        Toast.LENGTH_SHORT).show();
5        }
6    }
```

6.2.3.1 Control i gestió de la càmera

Per a la gestió de la càmera del dispositiu, OpenCV ens aporta dos codis font que són crucials per a la utilització i lectura d'aquesta.

En primer lloc, tenim la classe *JavaCameraView* la qual, primer de tot, ens ofereix la Vista que s'ha d'utilitzar per a poder incorporar la imatge de la càmera en la nostra interfície gràfica. Les Vistes, conegudes pel nom anglès *Views*, són tots aquells elements que en una aplicació Android permeten a l'usuari interactuar amb aquesta. Així doncs, aquí incorporem l'element que permetrà a l'usuari visualitzar imatges captades per la càmera.
El codi en *xml* per fer-ho és el següent:

```
1    <org.opencv.android.JavaCameraView
2        android:id="@+id/CameraView"
3        android:layout_width="match_parent"
4        android:layout_height="match_parent"
5        android:visibility="visible"
6        android:keepScreenOn="true"/>
```

La vista de la càmera és configurada de tal forma que ocupi tota la pantalla disponible (mitjançant el *"match parent"* en les seves mides) i mantingui la pantalla activa ("*keepScreenOn*"), ja que, les ulleres, en no detectar interacció, s'apagaven.

En segon lloc, ens ofereix la classe *CameraBridgeViewBase*, la qual actua com a controlador/gestor de la vista afegida a la interfície.
Per tal de poder començar a treballar amb ella, en inicialitzar l'aplicació (funció *onCreate()*), s'obté una referència a aquesta vista, mitjançant la qual fem visible l'element gràfic i posteriorment (funció *onResume()*) l'habilitem:

```
1    // camp en la definició de la classe
2    private CameraBridgeViewBase cameraBridgeViewBase;
```

```
1    protected void onCreate(Bundle savedInstanceState){
2        cameraBridgeViewBase = (JavaCameraView) findViewById(R.id.CameraView);
3        cameraBridgeViewBase.setVisibility(SurfaceView.VISIBLE);
4        cameraBridgeViewBase.setCvCameraViewListener(this);
5    }
```

```
1    protected void onResume(){
2        cameraBridgeViewBase.enableView();
3    }
```

En l'última línia que es mostra de la funció *onCreate()*, es pot veure com indiquem al controlador que l'activitat actual (*this*) actuarà de *listener*.
Per entendre millor això cal entendre els conceptes d'activitat i *listener*. Per una banda, una activitat en Android és l'execució que funciona com a gestora d'una pantalla de la interfície d'usuari. Entenent això, una aplicació que disposa de diverses pantalles d'usuari, disposa de les seves respectives activitats. Per altra banda, un *listener* és aquella activitat que rebrà la informació que un controlador obtingui de l'element que està sent controlat per ell.

El fet de ficar l'activitat principal com a *listener* ens modifica la declaració d'aquesta classe, que passa a implementar el *listener* necessari per a una vista de la càmera, el qual també ens el proporciona la classe *CameraBridgeViewBase*:

```
1    public class MainActivity extends AppCompatActivity
2            implements CameraBridgeViewBase.CvCameraViewListener2 {
```

A partir d'aquest *listener* cal implementar les 3 funcions que ens proporciona: *onCameraFrame*, *onCameraViewStarted* i *onCameraViewStopped*, d'entre les quals, la primera ens permet gestionar la imatge obtinguda de la càmera en frames, és a dir, en cada una de les instantànies que s'obtenen d'una gravació (una gravació és una seqüència d'imatges estàtiques o instantànies). Així doncs, aquesta, *onCameraFrame*, és la funció

on nosaltres introduirem el tractament d'imatges i, per tant, on en gestionarem la detecció.

Respecte a la càmera en el dispositiu que es fa servir en el treball, ens vam trobar amb un problema que ens va retenir durant un llarg període de temps: en obtenir la vista i afegir-la a la interfície gràfica es veia girada. Així doncs, tenint en compte que la primera classe vista, s'encarrega de la vista de la càmera i proporciona la seva configuració, vam incorporar codi en aquesta classe per tal de modificar la configuració existent i així mostrar la càmera de forma correcta. El resultat va ser l'esperat i vam aconseguir girar la imatge de la càmera, tot i això, la relació d'aspecte no era del tot adequada i, a més, posteriorment ens va perjudicar en la detecció d'objectes, ja que els frames tractats amb la detecció es perdien amb la nova configuració de la càmera.
Finalment, vam trobar una solució que no se centra en la càmera, sinó en la lectura dels frames que s'obtenen de la imatge gravada per aquesta. Així doncs, en la funció *onCameraFrame*, on es gestiona cada frame, vam introduir el següent codi:

```
1     public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame) {
2         Mat newFrame = inputFrame.rgba();
3         formatFrame(newFrame);
4     }
5
6     private void formatFrame(Mat frame) {
7         Core.flip(frame, frame, 0);
8         Core.flip(frame, frame, 1);
9     }
```

Com podem veure, mitjançant la classe *Mat* i *Core* que proporciona OpenCV, podem convertir el frame obtingut en una matriu amb els píxels de la imatge a color (línia 2) i, a aquest nivell, fer els moviments necessaris per girar el frame de la forma corresponent (línies 7 i 8).

Finalment, mitjançant la referència obtinguda al principi, alliberem la càmera un cop l'aplicació passa a segon pla o es tanca:

```
1     protected void onPause() {
2         if (cameraBridgeViewBase != null) {
3             cameraBridgeViewBase.disableView();
4         }
5     }
```

6.2.3.2 Detecció d'Objectes

OpenCV ens ofereix eines per a la detecció d'objectes, la qual es pot dividir en etapes segons la funció que tinguin aquestes:

Inicialització de la xarxa neuronal

El primer pas per inicialitzar la detecció d'objectes, és obtenir els fitxers de configuració de l'algorisme YOLO i inicialitzar la xarxa amb ella:

```
1     // camps en la definició de la classe
2     private Net net;
3     private boolean netInitialized = false;
4     private String yoloCfg;
5     private String yoloWeights;
```

```
1     private void initializeNet() {
2         net = Dnn.readNetFromDarknet(yoloCfg, yoloWeights);
3         netInitialized = true;
4     }
```

Una de les classes que ens ofereix OpenCV és *Net*, la qual ens permet crear i manipular xarxes neuronals artificials comprensibles. Per tal de generar la xarxa que controlarem amb aquest objecte, OpenCV ens proporciona un lector de models i configuracions de xarxes neuronals, *Dnn*. En el nostre cas, generem una xarxa neuronal emmagatzemada en Darknet, els fitxers de la qual, la configuració i el model de dades, tenim emmagatzemats en l'emmagatzematge intern del dispositiu, adreça que passem a la funció generadora.
Un cop fet això, ja tenim una xarxa neuronal configurada i preparada per funcionar.

Detecció d'Objectes

Per fer la detecció d'objectes un cop obtingut un frame és necessari seguir alguns passos:

1. Realitzar la detecció:
   a) Transformar la imatge en un objecte interpretable per la xarxa.
   b) Obtenir les capes que ha d'utilitzar la nostra xarxa neuronal per la detecció.
   c) Realitzar una passada per la xarxa neuronal per tal de fer la detecció.
2. Processar les deteccions:
   a) Obtenir els valors de les deteccions.
   b) Filtrar aquests valors segons un límit permès.

c) Si ha passat el filtre, obtenir informació necessària per poder dibuixar els quadres delimitadors i indicar les prediccions.

3. Filtrar les deteccions restants que puguin estar encavalcades.

Tots ells es poden desenvolupar gràcies a components que ens proporciona OpenCV en la seva llibreria.

Detecció

El codi per realitzar la detecció és el següent:

```
1     private List<Mat> detect(Mat frame) {
2         Imgproc.cvtColor(frame, frame, Imgproc.COLOR_RGBA2RGB);
3         Mat imageBlob = Dnn.blobFromImage(frame, 0.00392, new Size(416, 416),
4                                           new Scalar(0, 0, 0),false, false);
5         net.setInput(imageBlob);
6
7         List<Mat> result = new ArrayList<>();
8         List<String> outBlobNames = getOutputNames(net);
9         net.forward(result, outBlobNames);
10
11        return result;
12    }
13
14    private static List<String> getOutputNames(Net net) {
15        List<String> names = new ArrayList<>();
16
17        List<Integer> outLayers = net.getUnconnectedOutLayers().toList();
18        List<String> layersNames = net.getLayerNames();
19
20        for (Integer item : outLayers) {
21            names.add(layersNames.get(item - 1));
22        }
23        return names;
24    }
```

Com es pot apreciar, primer de tot (línia 2) adaptem el frame a processar perquè la seva gamma cromàtica es limiti als colors roig, verd i blau, i se'n genera un objecte binari (linia 3 i 4) que suposa una abstracció dels colors, mides i resolucions, així com algunes transformacions, del frame.
Un cop adaptat s'indica a la xarxa neuronal que aquell binari serà l'entrada a la xarxa quan aquesta s'executi (línia 5).
Abans d'executar, però, cal identificar les capes que requereix el model introduït a la xarxa, en aquest cas YOLO, per a processar les imatges. En la funció *getOutputNames()*

s'identifiquen les capes, que, per al model que nosaltres fem servir, se'n necessiten 3. Un cop obtingudes les capes, es pot executar la xarxa neuronal, la qual extraurà un resultat per a cada capa amb les seves respectives deteccions. Així doncs, en aquesta execució corresponen 3 resultats diferents de deteccions fetes.

Respecte a les capes de detecció i la seva importància en la xarxa neuronal, aquestes s'encarreguen de realitzar les deteccions per a diferents resolucions del frame, de tal forma que cada una s'encarrega de detectar elements de mides diferents: la capa de menys resolució s'encarrega dels elements més grans, mentre que la de més resolució s'encarrega d'elements més petits.

Processament

El processament dels resultats consisteix a obtenir informació de les deteccions de cada capa i fer un primer filtratge per a descartar aquelles deteccions amb poca confidència. El codi de la funció *processDetections()* és el següent:

```
1     private void processDetections(Mat frame, List<Mat> result,
2                         List<Integer> clsIds, List<Float> confs, List<Rect> boxes) {
3
4         for (int i = 0; i < result.size(); ++i) {
5             Mat level = result.get(i);
6             for (int j = 0; j < level.rows(); ++j) {
7                 Mat detection = level.row(j);
8
9                 Mat scores = detection.colRange(5, level.cols());
10                Core.MinMaxLocResult mm = Core.minMaxLoc(scores);
11                float confidence = (float) mm.maxVal;
12                Point classIdPoint = mm.maxLoc;
13
14                if (confidence > confThreshold) {
15                    int centerX = (int) (detection.get(0, 0)[0] * frame.cols());
16                    int centerY = (int) (detection.get(0, 1)[0] * frame.rows());
17                    int width = (int) (detection.get(0, 2)[0] * frame.cols());
18                    int height = (int) (detection.get(0, 3)[0] * frame.rows());
19
20                    int left = centerX - width / 2;
21                    int top = centerY - height / 2;
22
23                    clsIds.add((int) classIdPoint.x);
24                    confs.add(confidence);
25
26                    boxes.add(new Rect(left, top, width, height));
27                }
28            }
```

```
29          }
30      }
```

Com podem veure, realitzem iteracions per a recórrer els resultats (línia 4), o sigui, la sortida de cada capa, i per cada una, iteracions per tal de recórrer les deteccions que s'han fet en aquestes capes (línia 6).

Cada resultat és un objecte binari, del qual, cada fila n'és una detecció diferent. Per a aquestes, primer de tot obtenim informació sobre les prediccions (línies 9 i 10): quines són les classes amb més i menys probabilitat i on es troben (la seva posició) en el conjunt de totes les classes; i en guardem la informació de la predicció màxima (línies 11 i 12).
Amb aquesta, si supera el límit de confidència de predicció de classe establert a 50 % (línia 14), en calculem el respectiu quadre delimitador (línies 15 a 21): el seu punt central, l'altura, l'amplada i l'extrem on comença el quadre (punt superior esquerra).
Un cop obtingut tot, es guarda als seus respectius vectors la informació de la detecció i el quadre generat (línies 23 a 26).

Filtratge

El filtratge que es realitza aquí és l'últim pas de la detecció d'objectes, el qual se centra a organitzar els quadres delimitadors per tal d'eliminar els sobrants. Això ho realitza la funció *Dnn.NMSBoxes()* i el codi complet de tots els components que hi prenen part és el següent:

```
1   private void filterDetections(List<Integer> clsIds,
2                           List<Float> confs, List<Rect> boxes) {
3
4       int ArrayLength = confs.size();
5       if (ArrayLength >= 1) {
6
7           MatOfFloat confidences = new MatOfFloat(Converters.vector_float_to_Mat(confs));
8           Rect[] boxesArray = boxes.toArray(new Rect[0]);
9           MatOfRect boxesMat = new MatOfRect(boxesArray);
10          MatOfInt indices = new MatOfInt();
11
12          Dnn.NMSBoxes(boxesMat, confidences, confThreshold, nmsThresh, indices);
13
14          int[] ind = indices.toArray();
15          ArrayList<Detection> detections = new ArrayList<>();
16          for (int i = 0; i < ind.length; ++i) {
17
18              int idx = ind[i];
19              Rect box = boxesArray[idx];
```

```
20                  int id = clsIds.get(idx);
21                  float conf = confs.get(idx);
22                  int intConf = (int) (conf * 100);
23                  Point textPosition = box.tl().clone();
24                  textPosition.set( new double[]{textPosition.x, textPosition.y - 10});
25
26                  detections.add(new Detection(box, id, intConf, textPosition));
27
28              }
29              listener.setNewDetections(detections);
30          }
31
32      }
```

```
1       // MainActivity
2       public void setNewDetections(ArrayList<Detection> detections) {
3           synchronized (this) {
4               detectionsDone.clear();
5               this.detectionsDone.addAll(detections);
6           }
7       }
```

```
1       class Detection {
2           private Rect box;
3           private int id;
4           private int intConf;
5           private Point textPosition;
6
7           public Detection(Rect box, int id, int intConf, Point textPosition) {
8               this.box = box;
9               this.id = id;
10              this.intConf = intConf;
11              this.textPosition = textPosition;
12          }
13      }
```

Un cop obtingudes les deteccions finals amb la funció esmentada anteriorment, generem un objecte *Detection* per cada una per tal de guardar l'associació entre classe, probabilitat i quadre delimitador.
Després de generar un vector amb totes les deteccions, aquest s'envia a l'activitat principal (línia 29). Això ho fem, ja que l'activitat de detecció d'objectes és una activitat asíncrona a l'activitat principal on tenim l'obtenció dels frames, ja que, d'aquesta manera, evitem bloquejar la imatge de la càmera amb el processament d'objectes. Per

tal d'inicialitzar la tasca de detecció d'objectes, s'empra la següent línia de codi, en la qual podem comprovar com, en crear la tasca, hi passem l'activitat principal (*this* = *MainActivity*) així com la xarxa neuronal que farem servir per a totes les deteccions:

```
1     // MainActivity
2     objectDetectionTask = new ObjectDetectionTask(net, this);
3     objectDetectionTask.execute(newFrame);
```

En la segona línia, executem la tasca per al frame que s'estigui calculant en aquell moment.

Respecte als frames, com s'ha comentat en altres ocasions, no tots són processats: es fa un filtratge per tal de detectar quan un frame correspon a una seqüència d'imatges sempre iguals, per no fer deteccions de més, o d'imatges sempre diferents, per tal de fer deteccions de forma més sovint; o quan s'ha fet un canvi entre frames diferents a frames iguals, o a la inversa, per fer una nova detecció en detectar un d'aquests canvis. També, com s'ha comentat en la secció dels sensors, aquests condicionen la detecció o no de frames.

Per fer la comparació de frames i determinar quan són iguals i quan no, fem servir el següent codi:

```
1     public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame) {
2         Mat greyNewFrame = inputFrame.gray();
3         if (!oldFrame.empty()) {
4             areSimilarFrames = compareFrames(oldFrame, greyNewFrame);
5         }
6         greyNewFrame.copyTo(oldFrame);
7     }
8
9     private boolean compareFrames(Mat oldFrame, Mat newFrame) {
10        Mat difference = new Mat();
11        Core.compare(oldFrame, newFrame, difference, Core.CMP_EQ);
12        int count = Core.countNonZero(difference);
13        return count >= 15000;
14    }
```

OpenCV ens ofereix una funció per tal de comparar els frames i guardar les diferències en un objecte binari (línia 11), aquest objecte binari per cada píxel dels frames entrats que sigui igual, tindrà un valor 255, mentre que si no són iguals, ficarà un 0. Així doncs, seguidament contem quants números no són iguals a 0 per tal d'identificar quants punts semblants tenim en els frames (línia 12) i retornem que són iguals si supera els 15000 punts (línia 13). Aquest valor ha estat decidit després de fer diverses proves d'error, en les quals, aquest valor ha estat el més resultant.

7. Treball futur

Com bé s'ha comentat, el treball realitzat representa l'inici d'un projecte que té com a objectiu realitzar un estudi sobre com d'útil es l'awareness de les eines de detecció d'objectes. Així doncs, com a primer pas per dur a terme el projecte, s'ha realitzat la part de desenvolupament i configuració de l'eina que es pretén utilitzar per fer l'esmentat estudi. El següent pas a realitzar i que ens dóna lloc a un projecte futur seria establir un context específic on l'ús d'eines de detecció d'objectes amb smartglasses pugui resultar beneficiós per a la persona en termes d'awareness. Aquest pròxim pas pot suposar fer les modificacions pertinents de l'eina que s'ha realitzat en el treball presentat per tal d'adaptar tant el dispositiu com l'aplicació a l'estudi que es vulgui realitzar. Aquest estudi haurà de representar la part més important del pròxim treball amb una anàlisi minuciós del context escollit i del perquè es creuria que l'aplicació de les idees exposades podria millorar el context proposat.

Un cop s'hagi realitzat l'estudi esmentat, aquest obligarà a adaptar la base de dades de la detecció d'objectes per tal de ser més coherent amb les necessitats del context escollit. En aquest treball s'utilitza una base de dades genèrica que no aplica a un cas especific, ja que, no conté un gran ventall d'objectes de les diferents classes existents. Així doncs, aquest pas es limita a entrenar el model de dades desitjat mitjançant una xarxa neuronal (com podria ser la utilitzada al treball, Darknet).

Altres millores que es recomana realitzar a partir del treball aquí presentat i que podrien portar a un estudi extra, les quals no depenen tant del context escollit sinó que, poden influir en el correcte funcionament per a l'awareness, se centrarien en la configuració de l'aplicació. Actualment, s'han utilitzat certs valors que influeixen directament en la detecció d'objectes i que caldria calcular amb més precisió. Un exemple clar d'això en seria la comparació de frames similars o idèntics, la qual, si no es fa de forma correcta, pot suposar realitzar la comprovació d'objectes (càlcul molt exhaustiu) en frames innecessaris, ja que, segons el plantejament que hi ha per aquest cas, si els frames són semblants, significa que segueixen tenint els mateixos objectes i, per tant, es podria mantenir la detecció anterior, estalviant així una detecció extra. Altres paràmetres millorables de l'aplicació, seria la selecció de valors delimitadors per als sensors ambientals. Actualment, l'aplicació, mitjançant els sensors d'acceleració, detecta quan el moviment és accentuat o no, i d'aquest forma reduir el nombre de càlculs de frames realitzat. El que es vol aconseguir és establir valors exactes per poder decidir quan certs canvis provinents dels diversos sensors del dispositiu s'han de tenir en compte i, per tant, realitzar càlculs de frames o no i evitar càlculs innecessaris. Aquest estudi implicaria una millora del rendiment exponencial, ja que amb el conjunt de sensors treballant alhora s'evitaria realitzar un gran nombre de còmputs.

Com s'ha estat comentant als punts anteriors, el principal objectiu de la línia de treball proposada és fer un estudi de l'awareness. Per tant, la part més important serà quina informació mostrar i com a l'usuari. Aquesta informació no té per què ser només del tipus visual, també pot ser amb informació auditiva. Aquí deriva un estudi precís de

com fer arribar aquesta informació a l'usuari de forma no intrusiva per aconseguir allò que s'ha definit a un context principal.

Finalment, com a millora extra del conjunt del treball, proposem fer l'eina de detecció d'objectes portable. Ara mateix està configurada de tal forma que només es pot utilitzar en el sistema android de les Vuzix, però seria interessant poder traslladar l'eina a altres dispositius i que segueixi funcionant correctament sense que l'usuari hagi d'implicar-s'hi. Aquesta proposta té com a objectiu fer pensar en l'ampliació del context escollit i en l'actualització futura i progressiva del conjunt de l'entorn de treball (millorant amb els dispositius i tècniques).

8. Conclusions

Per concloure amb aquest treball volem comentar una sèrie d'aspectes que trobem rellevants respecte a diferents apartats que s'han comentat i reflexions que han sorgit arran de la realització d'aquest.

Pel que fa a la realització del treball, a mesura que avançava el temps de realització ens vam adonar que la línia de treball inicial marcada era molt llarga i ambiciosa respecte al període de temps disponible, i no ens donaria temps a realitzar tot allò que haguéssim volgut. Ens hauria agradat poder arribar a la segona fase de treball, aplicar l'Awareness, donat que era el principal objectiu d'ençà que vam iniciar aquest projecte. De totes maneres, estem satisfets d'haver pogut rectificar a temps i encarar des d'un altre punt de vista el projecte.

Respecte al mercat d'aquests dispositius ens vam emportar una gran sorpresa quan ens trobàvem sense forma d'adquirir o llogar un dispositiu. Vam poder constatar dos aspectes molt importants sobre les smartglasses: primer de tot, el mercat és car si es busca un dispositiu amb alta capacitat computacional; i el segon aspecte és la dificultat per adquirir un dispositiu, ja que les llistes d'espera arriben a ser d'un any, estan encara en desenvolupament o simplement no queden existències.

Ens hem adonat que per a la realització de projectes a curt termini (durada d'un semestre), amb solucions com les smartglasses, és recomanable que el dispositiu estigui adquirit o ja reservat, ja que, si no és així, la viabilitat és nul · la per la situació actual del mercat. Destacar també que el dispositiu emprat utilitza Android i per tant, el temps d'aprenentatge ha sigut relativament ràpid però, si se suma que l'entorn de treball és diferent, l'aprenentatge es pot tornar exponencial (depenent de les guies per desenvolupadors que tinguin les ulleres). Tot plegat, pot repercutir directament en el temps real de desenvolupament del treball.

Sobre la qüestió exposada de privacitat creiem que no hauria de ser un problema molt important. Pensem que si els desenvolupadors dels dispositius centressin una mica més de recursos en millorar aspectes de seguretat, com per exemple afegir alternatives als indicadors llum de gravació de càmera, seria més fàcil l'extensió d'aquest tipus de dispositius. És cert que, no tothom és conscient que s'ha de fer un bon ús d'aquestes tecnologies i s'aprofiten per treure'n benefici de qualsevol mena o simplement es viola la privacitat de les persones.

Aquest tema és complicat, ja que tot és a causa d'un component present en tots aquests dispositius: la càmera. El dilema és ben simple arribats a aquest punt, s'ha de limitar l'accés a aquests dispositius o s'ha d'oferir llibertat a tothom per poder adquirir-lo? Nosaltres creiem que és una tecnologia molt interessant, la qual proporciona molts avantatges per facilitar moltes tasques del dia a dia, però no creiem que sigui viable. Així doncs, s'ha de controlar, ja que pots estar, sense que ningú s'adoni, gravant i extraient informació (actualment amb les xarxes socials es pot saber qualsevol cosa d'algú)

pel teu propi benefici sense cap mena de regulació.

Per finalitzar amb aquest dilema exposat, volem fer referència a una frase d'Eugene Howard Spafford (1989), expert en seguretat informàtica:

"*L'únic sistema completament segur és aquell que està apagat, tancat en un bloc de ciment i segellat en una habitació envoltada de filats i guàrdies armats.*"

L'altra qüestió que s'ha plantejat mitjançant les proves realitzades amb el dispositiu, és fins a quin grau perjudiquen la salut dels ulls aquests dispositius. Mitjançant la cerca de treballs relacionats amb el tema de la salut visual i l'ús de smartglasses però, no s'ha pogut determinar amb proves suficients si realment afecten la salut ocular [21] [22].

Es comenta aquesta qüestió i seria quelcom interessant d'investigar en profunditat, ja que, fent proves, trobàvem que després de fer-ne un ús perllongat, l'ull s'ha de tornar a acostumar a una altra forma d'observar.

Més en relació amb el tema que ens porta a fer el treball, creiem que aquest tipus de dispositius, en temps reduïts, pot ajudar significativament en l'awareness, però un ús perllongat i regular, pot ser contraproduent i pot portar a la pèrdua d'atenció del món fora de les ulleres. Això, però, són suposicions les quals s'haurien de comprovar degudament amb un estudi exhaustiu.

Tot i afirmar que aquests dispositius poden ajudar en l'awarenes, creiem que l'elecció que nosaltres hem fet d'ulleres no és la més encertada per aplicar en un futur una solució d'aquest tipus, ja que, el fet de disposar només d'una pantalla i la forma en què està col · locada, et redueix el camp i profunditat de visió. De totes maneres, les Vuzix poden resoldre i complir amb les expectatives proposades de forma solvent.

Finalment, quant a la llibreria emprada per la realització del treball, és una eina molt potent i amb presència a moltes plataformes. S'ha de comentar, però, que aquesta llibreria presenta certs punts negatius, com la informació disponible per a desenvolupadors en Android. Durant el desenvolupament, s'ha trobat molta informació per Python, però per Android, trobàvem poca informació la qual estava desactualitzada o explicada de forma contrària a una altra font oficial (d'aquí ve el problema d'integrar OpenCV). A més, vam trobar que per les noves versions 4.x.x, no era possible integrar-les a Android i vam haver de recórrer a una versió anterior. Creiem que el nostre treball pot servir de guia per solucionar els problemes que hem experimentat per fer la integració.
Amb relació a això, cal destacar la llibreria creada per Google, *Tensorflow*, la qual té una documentació molt completa i millor per plataforma Android que OpenCV, segons el nostre criteri.

Vist l'apunt anterior, cal comentar que un cop feta la integració de la llibreria correctament, desenvolupar el codi per a la detecció d'objectes no resulta una tasca gaire complicada, ja que, com era d'esperar, la llibreria ofereix les mateixes eines per a codi Python que Android, així doncs, amb l'ajuda d'informació obtinguda per a ambdós

codis, es pot desenvolupar el codi per a la detecció de forma correcta.

Quant a la resta del desenvolupament, ens vam adonar que el filtratge de frames, tal com el volíem fer nosaltres, no era tan fàcil com semblava.

Per una banda, la detecció del moviment del cap va resultar un pèl difícil de configurar. Primer de tot, feia falta saber quin sensor identificava millor el moviment del cap, l'acceleròmetre o el giroscopi, ja que moure el cap no només implica pujar les ulleres, sinó inclinar-les. En segon lloc, calia especificar quins serien els barems que classificarien un moviment com a elevat o com a lleuger. Així doncs, es van haver de fer repetides proves amb moviments de cap reals per tal d'identificar aquests valors. Tot i que existeixen controladors del moviment més complexos, com els que s'utilitzen en sistemes de realitat virtual, creiem que aquesta solució és l'adequada per al sistema que nosaltres hem desenvolupat, però que cal tenir en compte que pot requerir canvis segons les condicions.

En segon lloc, la comparació segons similitud dels frames no és tan senzilla com sembla. Tot i que fem servir una eina que el mateix OpenCV ens ofereix, aquesta no ens resulta del tot precisa. En aquest sentit, hem trobat alguns problemes, que posteriorment solucionem amb temporitzadors de temps de detecció i refresc: si l'usuari mou el cap lentament de tal forma que no es detecta acceleració, entre un frame i el seu anterior la diferència és indetectable per l'eina de comparació, fent que es pugui recórrer un espai elevat amb el cap, amb diversos objectes nous a detectar, i que el sistema no se n'hagi adonat ni fet la corresponent detecció; si tenim les ulleres immòbils enfocant una situació o imatge en moviment, segons la llum, la resolució i la quantitat d'elements que hi hagi, l'eina de diferenciació no identifica els canvis, ja que un cop passada la imatge a gris (gamma cromàtica requerida per l'eina), no es distingeixen uns objectes d'altres i no es troben diferències. Així doncs, des del nostre punt de vista, l'eina que ofereix OpenCV no és la més adient si es necessita precisió en la comparació, i seria més adequat fer un estudi més exhaustiu sobre com fer una correcta comparació d'imatges [23].

Tot i les complicacions del codi, el resultat ha estat millor de l'esperat i n'estem molt satisfets, ja que hem aconseguit superar els inconvenients que ens han sorgit en el transcurs del desenvolupament i que ens feien ser poc optimistes amb el resultat final, com els problemes amb la càmera o el baix rendiment que teníem al principi.

Bibliografia

[1] *Definition of Awareness*. en. URL: https://www.merriam-webster.com/dictionary/awareness.

[2] Mica R. Endsley. "Situation Awareness Misconceptions and Misunderstandings:" en. En: *Journal of Cognitive Engineering and Decision Making* (feb. de 2015). Publisher: SAGE PublicationsSage CA: Los Angeles, CA. DOI: 10.1177/1555343415572631. URL: https://journals.sagepub.com/doi/10.1177/1555343415572631.

[3] *Extended Reality (Detailed Overview)*. en-US. Mayo de 2019. URL: https://smartglasseshub.com/extended-reality/.

[4] 42Gears Team. *6 Types of Wearable Technologies you must know- Wearable device trend*. en-US. Mayo de 2020. URL: https://www.42gears.com/blog/6-wearable-technologies-you-must-know-right-now/.

[5] Expert System Team. *What is Machine Learning? A definition*. en-US. Mayo de 2020. URL: https://expertsystem.com/machine-learning-definition/.

[6] James Chen. *Neural Network Definition*. en. URL: https://www.investopedia.com/terms/n/neuralnetwork.asp (visitado 18-10-2020).

[7] *Smart Glasses - Everything You Need To Know*. en-US. Jun. de 2019. URL: https://smartglasseshub.com/smart-glasses/.

[8] Dan Farber. *Google Glass ancestors: 45 years of digital eyewear (photos)*. en. Library Catalog: www.cnet.com. URL: https://www.cnet.com/pictures/google-glass-ancestors-45-years-of-digital-eyewear-photos/.

[9] *Future of AR smartglasses: How they will become the way we view the world*. en. Section: AR. Oct. de 2019. URL: https://www.wareable.com/ar/future-of-ar-smartglasses-7677.

[10] Munkh-Uchral Erdenebat, Young-Tae Lim, Ki-Chul Kwon, Nyamsuren Darkhanbaatar y Nam Kim. "Waveguide-Type Head-Mounted Display System for AR Application". en. En: *State of the Art Virtual Reality and Augmented Reality Knowhow* (mar. de 2018). Publisher: IntechOpen. DOI: 10.5772/intechopen.75172. URL: https://www.intechopen.com/books/state-of-the-art-virtual-reality-and-augmented-reality-knowhow/waveguide-type-head-mounted-display-system-for-ar-application.

[11] Paul Lee y Pan Hui. "Interaction Methods for Smart Glasses". En: (jul. de 2017).

[12] Philipp A. Rauschnabel, Jun He y Young K. Ro. "Antecedents to the adoption of augmented reality smart glasses: A closer look at privacy risks". En: *Journal of Business Research* 92 (2018), págs. 374 -384. ISSN: 0148-2963. DOI: https://doi.org/10.1016/j.jbusres.2018.08.008. URL: http://www.sciencedirect.com/science/article/pii/S0148296318303849.

[13] *Building an Object Detection Model from Scratch in Python*. Jun. de 2018. URL: https://www.analyticsvidhya.com/blog/2018/06/understanding-building-object-detection-model-python/ (visitado 19-10-2020).

[14] *Architecture Of CNN | CNN Image Recognition*. URL: https://www.analyticsvidhya.com/blog/2017/06/architecture-of-convolutional-neural-networks-simplified-demystified/ (visitado 19-10-2020).

[15] *Introduction to Object Detection Algorithms*. Oct. de 2018. URL: https://www.analyticsvidhya.com/blog/2018/10/a-step-by-step-introduction-to-the-basic-object-detection-algorithms-part-1/ (visitado 19-10-2020).

[16] Rohith Gandhi. *R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms*. en. Jul. de 2018. URL: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e.

[17] Mauricio Menegaz on. *Understanding YOLO | Hacker Noon*. URL: https://hackernoon.com/understanding-yolo-f5a74bbc7967 (visitado 25-08-2020).

[18] *Fotogramas por segundo*. es. Page Version ID: 129747505. Oct. de 2020. URL: https://es.wikipedia.org/w/index.php?title=Fotogramas_por_segundo&oldid=129747505.

[19] *OpenCV*. URL: https://opencv.org/about/.

[20] *OpenCV*. es. Page Version ID: 128322880. Ago. de 2020. URL: https://es.wikipedia.org/w/index.php?title=OpenCV&oldid=128322880.

[21] Natasa Herzog y Amer Beharic. "Effects of the Use of Smart Glasses on Eyesight". En: ene. de 2020, págs. 808-812. ISBN: 978-3-030-27927-1. DOI: 10.1007/978-3-030-27928-8_123.

[22] Bastian Stock, Tiago Patrick dos Santos Ferreira y Claus-Peter H. Ernst. "Does Perceived Health Risk Influence Smartglasses Usage?" en. En: *The Drivers of Wearable Device Usage: Practice and Perspectives*. Ed. por Claus-Peter H. Ernst. Progress in IS. Cham: Springer International Publishing, 2016, págs. 13-23. ISBN: 978-3-319-30376-5. DOI: 10.1007/978-3-319-30376-5_2. URL: https://doi.org/10.1007/978-3-319-30376-5_2.

[23] *Structural similarity*. en. Page Version ID: 973516356. Ago. de 2020. URL: https://en.wikipedia.org/w/index.php?title=Structural_similarity&oldid=973516356.

[24] *pjreddie/darknet*. en. URL: https://github.com/pjreddie/darknet.

[25] Nurizal Priandani, Herman Tolle y Fitri Utaminingrum. "Real Time Advanced Head Movement Recognition for Application Controller Based On Android Internal Gyroscope Sensor". En: *International Journal of Advances in Soft Computing and its Applications* 9 (abr. de 2017), págs. 70-87.

[26] Leif Oppermann y Wolfgang Prinz. "Introduction to this Special Issue on Smart Glasses". en. En: *i-com* 15.2 (ago. de 2016). Publisher: De Gruyter Section: i-com, págs. 123-132. ISSN: 2196-6826, 1618-162X. DOI: 10.1515/icom-2016-0028. URL: https://www.degruyter.com/view/journals/icom/15/2/article-p123.xml.

[27] Roope Raisamo, Ismo Rakkolainen, Päivi Majaranta, Katri Salminen, Jussi Rantala y Ahmed Farooq. “Human augmentation: Past, present and future”. En: *International Journal of Human-Computer Studies* (mayo de 2019). DOI: 10.1016/j.ijhcs.2019.05.008.